PatchMatch Belief Propagation for Correspondence Field Estimation and its Applications

Frederic Olivier Besse

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

October 2013

I, Frederic Olivier Besse, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Correspondence fields estimation is an important process that lies at the core of many different applications. Is it often seen as an energy minimisation problem, which is usually decomposed into the combined minimisation of two energy terms. The first is the unary energy, or data term, which reflects how well the solution agrees with the data. The second is the pairwise energy, or smoothness term, and ensures that the solution displays a certain level of smoothness, which is crucial for many applications.

This thesis explores the possibility of combining two well-established algorithms for correspondence field estimation, PatchMatch and Belief Propagation, in order to benefit from the strengths of both and overcome some of their weaknesses. Belief Propagation is a common algorithm that can be used to optimise energies comprising both unary and pairwise terms. It is however computational expensive and thus not adapted to continuous spaces which are often needed in imaging applications. On the other hand, PatchMatch is a simple, yet very efficient method for optimising the unary energy of such problems on continuous and high dimensional spaces. The algorithm has two main components: the update of the solution space by sampling and the use of the spatial neighbourhood to propagate samples. We show how these components are related to the components of a specific form of Belief Propagation, called Particle Belief Propagation (PBP). PatchMatch however suffers from the lack of an explicit smoothness term. We show that unifying the two approaches yields a new algorithm, PMBP, which has improved performance compared to PatchMatch and is orders of magnitude faster than PBP. We apply our new optimiser to two different applications: stereo matching and optical flow. We validate the benefits of PMBP through series of experiments and show that we consistently obtain lower errors than both PatchMatch and Belief Propagation.

Acknowledgements

I would like to thank my supervisor, Jan Kautz, for giving me the opportunity to pursue this PhD, and for his guidance throughout. I am grateful to Microsoft Research for having provided me with an invaluable scholarship, enabling me to meet the most knowledgeable people. My gratitude goes to Carsten Rother and Andrew Fitzgibbon, my supervisors from Microsoft Research, who provided me with the most valuable insight and advice, and gave me the opportunity to undertake an internship in their Cambridge labs.

Thanks to both Gabriel Brostow, my second academic supervisor, and Tim Weyrich, for having me as their teaching assistant in the Computational Photography and Capture course.

I would also like to thank Michael Bleyer and Christoph Rhemann for their advice and for sharing their research code with me.

Many thanks to my colleagues and friends in UCL: Fabian, Jozef, Maciek, Antonio, Malcolm, Yotam, Oisin, Tim, Gwyneth, Kazim, who made room 4.17 the greatest office to work in. Thank you for all the coffees, table football games, lunches, and endless conversations that we shared together. Special thanks to James, Fabrizio, Neill, Kartic and all the rest of the Kautzonauts.

I would like to thank my family: my mother, my grandmother and my sister for their endless encouragements.

Finally, my deepest thanks go to Thomy, who supported me during all these years. This would not have been possible without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Finding correspondence between images is a key problem that the machine vision field attempts to solve. Research in this area has been carried out for decades, resulting in many practical solutions which are commonly used in real-world applications, such as video editing for visual effects, medical image registration, object tracking for surveillance, object detection algorithms embedded in industrial robots and many more.

We can define the image correspondence field estimation problem as follows: given two images, the first being the target image and the second being the source image, we aim at retrieving for each pixel of the target image, a matching pixel from the source image, which represents the same physical point from the scene being projected onto both images. In other words, given two different projections of a 3D scene (the images), the aim is to find pairs of 2D points that correspond to the same 3D point. We can illustrate a correspondence field by using correspondence vectors, which represent the relative offsets of matching pixels between the two images, following a colour code, as shown in Figure 1.1.

While the human brain performs the task of image correspondence estimation effortlessly, designing an algorithm that can achieve comparable results while being efficient is a difficult task. When designing such algorithms, there are several considerations that are linked to the nature of the 3D world and its projection onto images that need to be taken into account. First, there is a strong appearance relationship that exists between two matching pixels. While there can be differences in illumination between the two images, affecting the appearance of the pixels, they often appear similar to the human eye. Second, there is also a strong coherency between neighbouring pixels. It is in fact common practice to consider *patches*, which are small windows of pixels in the image, instead of individual pixels, as the patch appearance is often related between the two views. Patches are also more distinctive than pixels. While it can be easy to match a single pixel in an image with many similarly looking pixels in the second image, which makes the matching ambiguous, patches tend to be more distinctive. Finally, as visible surfaces in the 3D world appear to be piecewise continuous, their 2D projections onto images also exhibit the same property, and therefore there is a certain smoothness level that is expected in correspondence fields.

In this thesis we consider two families of algorithms which can provide correspondence between images. The first class of methods is called Nearest Neighbour Search. Suppose that $\mathcal{U}$ is a set and D is a distance measure on $\mathcal{U}$, then for a given set $S \in \mathcal{U}$, the Nearest Neighbour Search consists in finding

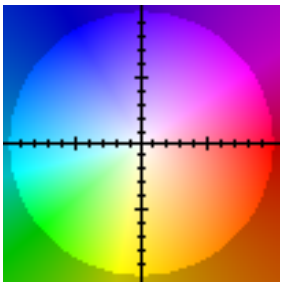


Figure 1.1: Colour representation of the motion vectors.

for an input query point $q \in \mathcal{U}$, and element $a \in S$ such that $D(q,a) \leq D(q,x)$ for all $x \in S$. In vision applications and specifically in our case, the query elements are often the patches of an image. Distance measures, also known as dissimilarity measures, are used to provide a quantitative measure of the similarity between the query elements and the most commonly used are described in further detail in section 2. The nearest neighbour is entirely defined by its distance to the query element. However, in vision applications this is not always ideal, as there are other considerations that need to be taken into account, such as the smoothness of the solution. This can be achieved by considering pairwise relationships between neighbouring patches for which correspondence is sought, as we explain below.

We consider another class of methods, which consists in minimising an energy function, crafted according to the problem that is being solved, with the assumption that the sought solution lies at the global minimum of this function. We assume that the correspondence field can be parameterised by a vector grid $\mathbf{u} = \{\mathbf{u}_s\}_{s=1}^{n}$ where s indexes *nodes*, typically corresponding to image patches, and $\mathbf{u}_s \in \mathbb{R}$ parametrises the correspondence vector at node s. We call $\mathbf{u}_s$ the *state* at node s. In this work we consider a special type of energy functions, which contain a *unary* and a *pairwise* term, and is defined as:

$$E(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{s=1}^{n} \psi_s(\mathbf{u}_s) + \lambda \sum_{s=1}^{n} \left[\sum_{t \in N(s)} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) \right], \tag{1.1}$$

with $N(s)$ being the set of *pairwise neighbours* of node s, ψ_s is the unary term, ψ_{st} is the pairwise term and λ controls the influence of the pairwise term. The unary term can be seen as a distance measure similar to that used in the Nearest Neighbour Search problem, while the pairwise term is a cost that defines an interaction between pairs of nodes, typically penalising any difference between their states and regularising the solution. The parameter λ is used to control the level of regularisation of the solution, and it is possible to reduce the problem to a Nearest Neighbour Search by setting $\lambda = 0$. One of the algorithms that can be used to minimise this type of energy is a message passing algorithm called Belief Propagation [Pearl, 1988].

In this thesis, we show that an efficient approximate Nearest Neighbour Search algorithm called PatchMatch [Barnes et al., 2009] shares similarities with the Belief Propagation algorithm. While the former excels in its computational efficiency, the latter allows for a more general form of energy to be minimised, comprising pairwise terms. We formally expose the link between these two algorithms and devise a new method, PatchMatch Belief Propagation (PMBP), which combines their strengths, and

show its benefits in two vision applications: Stereo Matching and Optical Flow.

In the following sections, we describe these two applications, and ways of validating algorithms that attempt at solving such problems. We then describe the common difficulties that arise due to the process of capturing images, and explain the importance of smoothness as well as the importance of computational efficiency, before stating our contributions in detail.

1.1 Applications

While there are many applications that are related to the estimation of correspondence between images, such as deblurring, high dynamic range imaging, and inpainting we focus in this thesis on two in particular: stereo matching and optical flow.

(a) Left image (b) Right image (c) Ground truth

Figure 1.2: Stereo matching. (a) is the left view, and (b) is the right view. The stereo matching application consists in finding the horizontal offsets (assuming that the images have been rectified) between matching pixels of the two views. (c) is a representation of the ground truth disparity map, where large disparities are represented in brighter colours while small disparities are in darker colours.

The stereo matching task consists in finding dense correspondences between two images that were acquired in a stereo setup. In this configuration, two cameras are placed next to each other and capture images at the exact same time. It is analogous to the human vision system, and therefore the two images are called left and right views. In a perfect setup, the correspondences between images are located on the same scan lines. Consequently, there is only one degree of freedom in the search, which is the offset along the x axis. This offset is called the disparity. Depth can be retrieved from the known disparity, as in this particular settings disparity and depth are inversely proportional. The interest in stereo matching stems from the fact that it is a simple way to produce a sensation of depth without having to generate a 3D environment, taking advantage of the way that the human brain converts this horizontal offset, the disparity, into depth information, amongst other cues. In practice it is difficult to obtain a perfect stereo setup consisting of two cameras having the same orientation and whose positions differ by a horizontal offset only. However, a processing step called rectification can be run on the images to convert them into two new images where the scanlines are aligned. To understand this mechanism, we need to define the basis of epipolar geometry.

Let us consider two cameras, C_0 and C_1 and their respective centres c_0 and c_1. The cameras

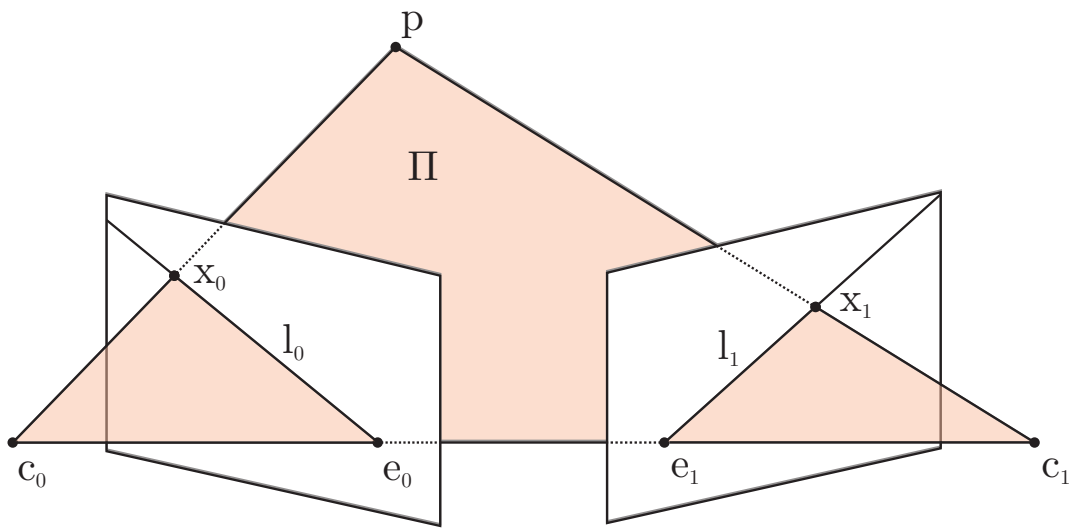


Figure 1.3: Epipolar geometry of a multi-camera scene.

produce one image each, I_0 and I_1. A pixel x_0 in I_0, projection of a point p in space, as seen in figure 1.3, projects to an epipolar line segment l_1 in I_1. This segment, on I_1 is bounded by two points: the first is the projection on I_1 of the camera center c_0, which is called the *epipole* e_1. The second point is the projection of the original ray at infinity, p_∞. The extension of the epipolar segment is known as an *epipolar line*. Projecting an epipolar line in the other image results in a corresponding epipolar line, and the two form an *epipolar plane* Π which passes through the interest point p and both camera centres. Rectification is a warping process which consists in transforming the images such that the horizontal scanlines correspond to epipolar lines [Hartley and Zisserman, 2000]. Working with rectified images makes the stereo matching task easier as it restricts the search to horizontal lines, which is often seen as a 1D correspondence field.

(a) First image (b) Second image (c) Ground truth

Figure 1.4: Optical flow. (a) is the target image, and (b) is the source image. Optical flow computation consists in finding the motion vectors that explain how each pixel of image (a) evolved in image (b). (c) is a representation of the ground truth motion vectors.

The optical flow application is more general than stereo matching, as the setup is not constrained. It is generally thought as the estimation of an independent 2D motion at each pixel. There is no restriction on the capture of the images, as they are usually not captured at the same moment in time (contrary to

stereo matching). This translates in possible motions of the objects in the scene from one image to the other, which makes the task less constrained and therefore more difficult than stereo matching. It is often used on consecutive pairs of images extracted from a video. It is a low level application that is often part of the core of other methods, such as motion tracking [Decarlo and Metaxas, 2000] or video editing [Zitnick et al., 2005]. An illustration can be seen in figure 1.4, as well as the typical colour representation of the motion vectors.

1.2 Validation

As with any scientific method, the process of designing and evaluating an algorithm goes through a validation step. This often consists in knowing the ground truth solution of several cases, and providing a measurement of the variation of the result produced by the tested algorithm with the ground truth. It typically takes the form of an error measure of the produced output. There are therefore two important steps to consider for the validation: the first is the production of the ground truth data, and the second is the conception of the error evaluation function for a given output.

There are several ways of creating data with ground truth correspondences. For ground truth displacement fields, a first option is to generate synthetic data, created from 3D modelling software which allows the generation of the ground truth correspondence values, such as [Butler et al., 2012]. Another way is to use extra cues such as hidden fluorescent textures [Baker et al., 2011], or another capture device such as laser scanner devices [Geiger et al., 2012]. Another approach is to acquire high resolution images, mixed with some manual labelling of planar surfaces and direct alignment techniques of these surfaces [Scharstein and Szeliski, 2002]. In the frame interpolation application, which consists in synthesising the frame that is located temporally in-between two frames of a video, one can simply use a high speed camera to retrieve the ground truth image and use the adjacent pictures as testing pair.

After having created a test set with ground truth correspondences available, one needs to produce an evaluation measure to quantitatively evaluate the methods. When the ground truth takes the form of a displacement field, a typical measure is a norm of the error between the ground truth field and the candidate field. For Stereo Matching, the Root-Mean-Squared (RMS) error measured in disparity unit is often used which is a simple L2 norm on the vectors formed of the ground truth displacement and the candidate displacement. Another option is to calculate the percentage of bad matching pixels, where a bad matching pixel is defined by a pixel whose estimated disparity differs from the ground truth disparity value by more than a certain disparity threshold [Scharstein and Szeliski, 2002]. For Optical Flow applications, two measures are often used: the Angular Error (AE) and the Endpoint Error (EE or EPE) which is also related to the L2 norm, similarly to the RMS used in stereo [Baker et al., 2011]. For frame interpolation applications, one can simply calculate the RMS between the synthesised and the ground truth images.

We isolate two important algorithmic properties that are beneficial for applications such as stereo matching and optical flow: computational efficiency and smoothness.

1.3 Computational Efficiency

A desirable property of algorithms embedded in real world vision applications is efficiency. Such algorithms are often required to perform in real time, meaning that when applied on a pair of consecutive images of a video, the solution for a frame needs to be computed before the next frame is captured. Video streams range from typically 20 to 120 frames per second (FPS). This means that a full computation must be run in 8 to 50 milliseconds depending on the requirements. When not able to perform in real-time, it is an advantage to be able to run the algorithm at interactive speeds, meaning that the effect of any changes in the input or the settings can be visualised quickly enough to produce an interactive feedback. This prohibits the use of computationally expensive methods, and a tradeoff must often be made between quality and speed. Nearest Neighbour methods often introduce suitable data structures used to accelerate this search, as explained later in section 2.

In this thesis, we focus on a recent technique, called PatchMatch [Barnes et al., 2009] which can be classified as an approximate Nearest Neighbour technique, and that uses the local coherency present in natural images to accelerate the search. It is a randomised algorithm that proceeds by first assigning correspondence between patches at random. After this initial step, good matches are refined by performing a random search around their positions, and they are also propagated to neighbouring pixels. The propagation mechanism is the key idea of PatchMatch, and it is based on the assumption that when considering two images, the correspondence field is likely to be piecewise constant, which is similar to the smoothness assumption mentioned earlier, but used to accelerate the search when seeking nearest neighbours of sets of contiguous pixels or patches.

This algorithm has been successfuly applied to applications that require real-time feedback, and is described in further detail in Chapters 2 and 3. Correspondence fields output by PatchMatch often exhibit clusters where the solution is almost constant on each cluster, for example when parameterising the match field with offsets. This phenonemon, refered as the *coherence of the mapping* in [Korman and Avidan, 2011] is a byproduct of the propagation step. However, nothing prevents these clusters to be broken down if a better solution is found at individual pixels, which is why we will refer to this property as the *implicit* smoothness of the solution field. In other words, the smoothness assumption is used in this algorithm as a way of generating new candidate matches, which biases the solution in being piecewise constant. However, it lacks the desirable property of including an *explicit* smoothness term, where smoothness is actively sought by penalising non smooth solutions.

1.4 The Importance of Smoothness

It is a common and fair assumption to say that the world is continuous and displays local coherency as is it largely made of surfaces. Images taken of the world reflect this coherency and vision applications in turn incorporate this idea as a smoothness prior on the sought solution. Estimating correspondence between images is a difficult and ill-posed task and therefore such priors can help reducing the ambiguity that exists. In this section, we expose several factors which are mainly linked to the process of capturing images that contributes to the complexity of vision applications, and explain how the smoothness

(a) Clean image (b) Noisy image

Figure 1.5: Illustration of noise. (a) shows a noise-free version of an image, while (b) show the same image with 10% of Gaussian noise added.

assumption is beneficial.

Noise Mainstream cameras usually embed either a Complementary Metal Oxide Semiconductor (CMOS) or a Charge-Coupled Device (CCD) sensor in order to acquire a digital picture. The photons hitting the sensors are converted to electrons and then to bits. Thus, during the process of acquiring an image, the signal undergoes several transformations that alter it, and are source of different types of noise, which can be defined as a random variation of brightness or colour on the image. There are several types of noise that typically corrupt the digital image. The first one is the *Fixed Pattern Noise*, which is caused by the variance over the collection sites, or in other words, by the fact that each cell that collects the light is slightly different and does not respond in the exact same way to the same input. The second kind is the *Dark Current Noise*, and is related to the thermal energy. The environment temperature for example adds noise to the image and is one of the causes of dark current. The third kind is the *Shot Noise*, that is a consequence of the quantum nature of light. There also several extra related steps that can affect the quality of the image. The signal undergoes a white balancing operation, in order to attempt to generate colour images that are consistent with human perception, and then a gamma correction to prepare the images to be displayed on a monitor. This is an artefact that we find in most images. An illustration of a noisy image is given in figure 1.5. The process that aims at removing the noise from a picture, by separating the underlying image from the corrupted noise layer, is called denoising. However it is commonly preferred to employ an algorithm that is somewhat robust to noise while estimating correspondence, instead of explicitly denoising the image, which might lead to a further loss of information as it is difficult to perfectly separate signal from noise.

Quantisation When acquiring an image, an analog to digital conversion is done, in order to digitise the signal. A digital camera encodes values with typically 8 to 16 bit range per channel, yielding 256 to 65536 possible different pixel intensities. The process of turning a physical value such as real world radiance into a precision-limited number (that can be stored in a computer) is called quantisation. Encoding the radiance of a pixel as an 8 bit intensity is an approximation of the real physical value. Thus,

(a) Clean image (b) Blurred image

Figure 1.6: Illustration of blur. (a) shows a blur-free version of an image, while (b) show the same image with artificial motion blur added.

precision and accuracy will be lost during digitalisation. For example, we can think of a scene where radiance values are contained in the range [0,5]. This range, if not quantised, is capable of revealing the finest details to a suitable sensor (the human eye is a good example), since it can contain an infinite number of different values. However, integer quantisation will map those to only 6 different values (0, 1, 2, 3, 4, 5). This does not allow much detail in the final image, and this is the main problem raised by quantisation. This is especially a problem when the scene is either too dark or too bright. These extremes illumination conditions can also lead to under- or over-saturated areas, where the intensities take the maximum, or minimum value allowed by the sensor, leading to a loss of detail and distinctiveness.

Motion Blur During the image acquisition process, the shutter of camera stays open for a certain amount of time to allow enough photons to hit the sensor. This interval is regulated by a setting of the camera that is known as the *shutter speed*. The capture performed by the sensor can be seen as in integration of the continuous light entering the lens. When objects in the scene are moving however, this integration can result in blur if the shutter stays open too long. Both images of the same scene can have different amount of blur, and it might be difficult to accurately match blurry areas, which are also less distinct as they often appear smoother. A field of computer vision deals with deblurring images, however, similarly to the noise problem, most correspondence field estimation technique aims at being robust to a certain amount of blur.

Occlusions and disocclusions When capturing two images of a scene, the objects present will not appear be in the same position in image space between the shots. This change in position is affected by both the depth of the object, and naturally the independent motion of the object which can happen if the images are not captured simultaneously. This will lead to both occlusions of previously visible areas, and also disocclusions of previously hidden regions. This means that the image correspondence field estimation is not a one-to-one mapping from one image to the other, as some pixels in the target image will have no match in the source image, and inversely. This implies that successful matching algorithms need to be able to handle these cases, and sometimes extrapolate the position of the hidden pixels in

(a) Target image (b) Source image

Figure 1.7: Illustration of different problems due to motion. On this pair of images, we can notice several factors that make the optical flow process difficult. First, the table surface is highly reflective, and because of the change of position of the camera between the two poses, different regions of the table are highlighted. Furthermore, shadows cause a significant change in the appearance of the table patches which further complexifies the matching process. Finally, occluded and disoccluded areas of the table will not have any true match and need to be taken care of during the correspondence field estimation.

either image, depending on the application and the required behaviour.

Motion Finally, we consider another difficulty related to the previous one, caused by moving objects. Methods relying on patches are the most affected, as motion changes the appearance of patches containing simultaneously multiple objects with independent motion. The typical example is a patch containing a foreground and a background object, with the foreground object moving and the background being fixed. This produces a change in a local area of the patch and that brings ambiguity to the problem. Often, matching algorithms associate patches with the surface that is supporting the center pixel of the patch. Several techniques exists to address this problem, which often consist in estimating the support region of the foreground object that the center pixel lies onto. Another problem is the change in appearance in the image space of objects under different poses due to the projective process of image capture. Finally, other factors further complicate the process, such as change in illumination, reflective surfaces, and shadows to name a few. Some of these factors are illustrated in figure 1.7.

The artefacts involved in image capture that we have seen above naturally yield a degree of ambiguity in the matching process. When the patch appearances are corrupted, an algorithm relying only on appearance, such as a nearest neighbour search algorithm, might associate a wrong patch which appears to be more similar to the sought target patch than the actual ground truth. Therefore, it is crucial to introduce prior knowledge into the matching process, such as a level of smoothness in the correspondence field. This can be done naturally within an energy minimisation framework by using a smoothness prior, such as the pairwise term of equation 1.1. The Belief Propagation algorithm, described in detail in Chapters 2 and 3 is a well-established technique that can be used to minimise such energies. However, its computational cost is high, especially when the search space is high dimensional, or large.

1.5 Problem Statement

In this work we address the image correspondence problem by posing it as a continuous labelling problem which we solve in an efficient way. PatchMatch is a successful algorithm that can be used to efficiently find correspondence between images in a continuous space, however it lacks the presence of an explicit pairwise term. Belief Propagation is a well-known algorithm that can optimise energies containing both unary and pairwise terms, which is the property missing in the PatchMatch formulation, however it is computationally expensive and very non-efficient in continuous spaces. We can therefore summarise the research described in this thesis with the following question: **"Can we link PatchMatch and Belief Propagation and combine the efficiency of PatchMatch with the more generic formulation of Belief Propagation to optimise an energy containing a regularisation term that encourages smoothness?"**. We demonstrate the use of this algorithm to two applications: stereo matching and optical flow.

1.6 Contributions

The contributions of this thesis are:

- Formulation of PatchMatch and Belief Propagation under the same algorithmic framework, exposing for the first time the link that exists between them
- Creation of a novel algorithm, PatchMatch Belief Propagation (PMBP), which combines the strength of two known algorithms (namely PatchMatch and Belief Propagation) to address their weakness and produce superior results while being efficient and able to optimise an energy containing a regularisation term that encourages smoothness
- An open source implementation of PMBP with a comprehensive structure allowing for easy extension to new applications
- Application of PMBP to the stereo matching problem and creation of a realistic pairwise term for a planar surface model of the scene
- Extension of PMBP to the optical flow estimation problem with an over-parameterised affine model
- Creation of a geometrically based optical flow algorithm in a high dimensional space using PMBP as optimisation method

1.7 Thesis Structure

Chapter 2 contains the literature review of the different fields that are related to our work, including Image Matching, PatchMatch, Belief Propagation, Stereo Matching and Optical Flow methods.

Chapter 3 presents our main optimisation algorithm: PMBP which combines the efficient propagation step of PatchMatch with the capability of optimising general energies containing pairwise terms of

Belief Propagation. Additionally, it includes a formulation of PatchMatch and Belief Propagation under a common framework, exposing an existing link between them.

The next three chapters are applications based on our algorithm PMBP. Chapter 4 presents a stereo matching application using a plane model at each point combined with a novel pairwise term to encourage surfaces to have similar positions and normals. Chapter 5 first presents a fast optical flow algorithm based on PatchMatch and its practical application to fast video editing, followed by the introduction of PMBP to address certain weaknesses as well as an over-parameterised formulation using an affine motion model. Finally, Chapter 6 describes a novel geometrically based optical model which uses a high-dimensional space, that can be optimised with PMBP due to its efficiency.

Chapter 7 closes this thesis with conclusions and a discussion about potential future work.

Chapter 2

Literature Review

Finding correspondences between two images is an active research problem that applies to many fields. With technology and computational capabilities continuously increasing, the challenge of finding reliable correspondences in a reasonable time still remains, while low computational time is crucial for providing concrete solutions to many real-world applications. In this chapter, we first give a brief overview of the most common similarity measures as they are a core part of correspondence estimation techniques using images. Then, we review the state of the literature in correspondence estimation as well as general acceleration techniques. One of them is called PatchMatch, which is one of the two main algorithms that our work is based on. We dedicate a section to describe more in details various methods that use the PatchMatch algorithm. The other technique that is used thoroughly throughout this thesis is Belief Propagation. We focus in particular on Belief Propagation for continuous spaces. Finally, we present two specific imaging applications and their related work: stereo matching and optical flow estimation.

2.1 Similarity measures

Every correspondence field estimation algorithm uses a certain similarity measure to quantitatively evaluate a proposed correspondence. It it useful to discriminate between *similarity* and *dissimilarity* measures. The former gives a score of how similar the two elements are with each other, while the latter gives an error which is often seen as a cost in energy based frameworks. In this section we describe briefly the following measures: Sum of Squared Differences (SSD), Sum of Absolute Differences (SAD), Normalised Cross Correlation (NCC) and Mutual Information (MI).

The Sum of Squared Differences (SSD), when applied between two images, computes a difference in intensities between them by squaring the pixel-wise differences and summing them. It is a quadratic measure and is the square of the L2 norm. It is formally defined as:

$$SSD(A, B) = \sum_{i=1}^{N} (A_i - B_i)^2, \tag{2.1}$$

where A and B are the two images (or regions of interest) being compared and N is the number of pixels in each one of them. Because of the quadratic term, it is very sensitive to outliers. This measure is sometimes averaged with the number of pixels compared, and is in this case called Mean of Squared Error (MSE). This family of measures can also be used alongside a truncation term τ to make it more

robust to outliers:

$$SSD_{\text{trunc}}(A, B, \tau) = \sum_{i=1}^{N} \min((A_i - B_i)^2, \tau), \tag{2.2}$$

where N is the number of pixels in the region of interest.

The Sum of Absolute Differences (SAD) is based on the L1 norm and is less sensitive to outliers contrary to its quadratic counterpart seen above. It is defined as:

$$SAD(A, B) = \sum_{i=1}^{N} |A_i - B_i| \tag{2.3}$$

and it can also be truncated for increased robustness:

$$SAD_{\text{trunc}}(A, B) = \sum_{i=1}^{N} \min(|A_i - B_i|, \tau). \tag{2.4}$$

Note that these similarity measures are also commonly applied to the gradients of the images, instead of only raw intensities. It is commonly seen in correspondence field estimation applications such as Stereo Matching [Kanade et al., 1995].

The Normalised Cross Correlation (NCC) measure evaluates the similarity between two images in a normalised way, in order to add robustness to change in lighting and brightness, which is well suited to imaging applications, as it encourages matching the underlying patterns instead of the raw intensities. It is defined as:

$$NCC(A, B) = \frac{\sum_{i=1}^{N}((A_i - \bar{A}) \cdot ((B_i - \bar{B}))}{\sqrt{\sum_{i=1}^{N}(A_i - \bar{A})^2 \cdot \sum_{i=1}^{N}(B_i - \bar{B})^2}}, \tag{2.5}$$

where $\bar{A}$ and $\bar{B}$ are the mean intensities of images A and B respectively. This measure returns a correlation value, so that if the images are identical, their NCC is equal to 1. It has been successfully used in areas such as template matching [Lewis, 1995].

Finally, the Mutual Information (MI) measures the amount of information that one variable contains about another. While it originated from the information theory field, it is now commonly used in imaging and was introduced as a similarity measure between images by [Viola and Wells III, 1997] and [Maes et al., 1997]. Mutual Information is related to the joint entropy of a pair of variables. To define it, we first define the entropy of a variable and the joint entropy of two variables. The entropy of a variable A is defined as:

$$H(A) = \sum_{a} p_A(a) \log p_A(a), \tag{2.6}$$

where p_A is the probability function of a, and, given two variables A and B, their joint entropy is then defined as:

$$H(A, B) = \sum_{a,b} p_{A,B}(a, b) \log p_{A,B}(a, b). \tag{2.7}$$

where $p_{A,B}$ is the joint probability function of a and b. Finally, the Mutual Information is calculated as follows:

$$MI(A, B) = H(A) + H(B) - H(A, B), \tag{2.8}$$

which translates into:

$$MI(A,B) = \sum_{a,b} p_{A,B}(a,b) \log \frac{p_{A,B}(a,b)}{p_A(a)p_B(b)}. \tag{2.9}$$

Intuitively, it favours finding the most complex regions (maximising individual entropies) that explain each other as well as possible (minimising the joint entropy).

2.2 Correspondence in Images

In this section we review common techniques that can be used to find correspondence in images. These include Nearest Neighbour search methods as well as image descriptors.

Nearest Neighbour (NN) search consists in finding, for a point in a given space, the closest point under a distance function. In imaging applications, points are often pixels, patches or features and the metric used is one that can be applied to these elements, such as the similarity and dissimilarity measures described in the previous section. Here, we focus on NN Search techniques that can be applied to image patches or features in particular, as these are the main types of elements that we use. We will see later in this thesis that while finding the closest element based on its appearance is an important component of correspondence field estimation algorithms, there are some other considerations to take into account in this task, such as the smoothness of the sought solutions. However, Nearest Neighbour search techniques are still crucial components to many computer vision applications.

A certain family of techniques that can be used to perform Nearest Neighbour search are tree-based acceleration techniques which used various tree data structures applied on data in order to accelerate the search queries. KD-trees [Friedman et al., 1977] are very common in the field. A KD-tree is a binary tree where each node corresponds to a split of the space. This partitioning is adapted to the data. It works as follow: given a set of points of dimension d, building the trees starts with computing a hyperplane that splits the space and which is orthogonal to a chosen dimension. This hyperplane is often chosen by calculating the median value of the points within the dimension that has the greatest variance. The same process is repeated recursively until each leaf typically corresponds to a point of the data set. The height of the tree is then $\log_2(N)$ where N is the number of points. Consequently, each subsequent query requires $\log_2(N)$ comparisons to give an answer. However this method alone might not return the actual nearest neighbour, and therefore it is usually followed by a backtracking operation to find better candidates, which is a step often performed by using priority search [Arya and Mount, 1993]. To reduce the computational cost, an approximate search can be set up by limiting the number of nodes searched. The efficiency of the KD-tree decreases for very high dimensional data, because of the backtracking step. Dimensionality reduction methods such as Principal Component Analysis (PCA) [Jolliffe, 2005] are commonly used in conjunction with KD-trees to further accelerate the search [Sproull, 1991] and are called PCA-trees. Variations of KD-trees are also commonly used, such as Randomised KD-trees [Silpa-Anan and Hartley, 2008] which build multiple KD-trees with the same data to accelerate the search and Random Projection trees (RP-trees) [Dasgupta and Freund, 2008] which use random splits of the space. There are also various other tree structures that exist in the literature. Ball-trees [Omohundro, 1989] use a construction process that is similar to the k-means method

[MacQueen et al., 1967]. Vantage Point trees (VP-trees) [Yianilos, 1993] store a point and a radius at each node. Under one child are all the points that are within the radius of the parent's point, and on the other child are the rest. Tree-Structured Vector Quantization (TSVQ) [Wei and Levoy, 2000] which is commonly used for data compression is a structure that is trained from a dataset and generates a binary-tree-structured codebook, which is then used for fast Nearest Neighbour queries.

Another type of method to consider for establishing correspondence are those which generate points of interest in the images. These points are often distinctive and salient enough to be able to persist through changes in pose and illumination, and can therefore be easily matched. These can be used in conjunction with the tree structures mentioned above to accelerate the search [Mikolajczyk and Matas, 2007]. One approach is to use a corner detector which are often distinctive and present in both images. The Harris corner detector has been widely used for this purpose [Harris and Stephens, 1988]. [Lowe, 2004] revolutionised the field of machine vision with the Scale Invariant Feature Transform (SIFT) algorithm and the SIFT feature points. These features display many interesting properties such as invariance to scaling and rotation, partial invariance to change in illumination, camera view point and show high distinctiveness. The first step of the method consists in detecting local extrema in a difference of Gaussian scale space, which correspond to the key points positions. A scale and orientation are assigned to each point, and the descriptor is then computed, by measuring the local image gradients. There are several other methods that detect points of interest and are commonly used in the field, such as PCA-SIFT [Ke and Sukthankar, 2004], Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], Speeded Up Robust Features (SURF) [Bay et al., 2006], and steerable filters [Freeman and Adelson, 1991]. The main drawback of these methods is that they give only sparse correspondence, which is often not suitable when looking for a dense field of matches.

Finally, we focus on a recent algorithm called PatchMatch [Barnes et al., 2009] which performs a Nearest Neighbour search in an efficient way. The authors introduce a fast way to establish matches between patches using a mixture of random sampling and natural coherence in an image. While common tree-based acceleration methods provide an accelerated search, the computational time is often not low enough to allow real-time interactions, and the memory used can increase substantially. Instead, PatchMatch proposes an incremental way to find matches, based on the assumption that although one random choice will probably not give a good corresponding match, a large number of random assignments will be more likely to result in at least one good match. Another assumption of the PatchMatch algorithm is that, given the natural structure of images, if one patch is assigned a good match, then we can propagate this match to the neighbouring pixels. Thus, the randomised algorithm starts with an initial guess, and alternates between two phases: propagation which spreads good matches to adjacent pixels, and random search, which looks for better matches randomly in a given area around the current position. The performances of PatchMatch offer real-time interaction, especially with the GPU implementation, which can be 20 to 100 times faster than the previous state of the art algorithms. This is described in more details in Chapter 3 as it is a core component of this thesis. While the algorithm is an efficient Nearest Neighbour search method, it lacks a fundamental property: an explicit smoothness term which is very often needed

for machine vision applications.

2.3 PatchMatch Algorithm Related Literature

The PatchMatch algorithm has been actively used in the literature since its original publication by [Barnes et al., 2009], as it is an efficient way to retrieve a reasonable nearest neighbour solution when applied to images, due to their strong local coherency. In this section we review the methods that attempt at improving PatchMatch, or use it as part of their core algorithm. We put the emphasis on the presence -or absence- of smoothness in the methods that we review.

[Barnes et al., 2010] extend PatchMatch in their Generalized PatchMatch algorithm, where several improvements are made. First, while PatchMatch was performing a simple nearest neighbour search, this new method can be used to retrieve k nearest neighbours, by storing the matches in a heap data structure. The search is also extended to multiple scales and rotations, which are evaluated *on-the-fly*. An important advantage compared to other algorithms performing nearest neighbour search is parameters such as rotation and scale can take any value within a specified range. On the contrary, other algorithms such as KD-trees, require the different scales and rotations to be discretised in order for the transformed patches to be inserted in the tree. The Generalized PatchMatch does not need any discretisation as the data comparison is performed *on the fly*. The propagation step is also altered compared to the original PatchMatch in order to reflect the different orientations and scales of the matches. A new step, called enrichment is added, which consists in propagating candidates matches not only spatially, but also through corresponding matches. For example, if patch B is a good match to patch A, and patch C is a good match for patch B, there is a change that patch C will be a good match for patch A. In other words, it uses the nearest neighbour's nearest neighbours as source of other candidates. The authors show that their algorithm can be used with arbitrary descriptors and distances, and is not limited to the SSD measure only. They also propose an alternative parallelisation scheme where the image is divided in several horizontal bands which are processed on separate cores, with a synchronisation step ensuring the propagation through adjacent tiles. Finally, they propose various applications that can benefit from their method, such as denoising, clone and symmetry detection and object detection using SIFT descriptors. A noteworthy conclusion of their work concerns the denoising, where they remark that their algorithm gives a PSNR worse than that of the non local denoising method by [Buades et al., 2005], illustrating that their algorithm finds better matches and therefore is matching the noise in the image. This can be interpreted as the natural consequences of not having an explicit smoothness term within the optimisation which can be beneficial for applications where the smoothness prior is strong, such as denoising.

Coherency Sensitive Hashing (CSH) [Korman and Avidan, 2011] combines PatchMatch and Locality Sensitivity Hashing (LSH) [Datar et al., 2004] into an efficient algorithm that is three to four times faster than PatchMatch when performing nearest neighbour search. The authors point out that the key step of PatchMatch is the propagation which is done in a spatial manner, and use a hashing scheme similar to the one used in LSH in order to propagate matches to patches that have been hashed to the same values. The hashing is appearance-based, which means that this new propagation is done in appearance space. In addition, this hashing is also used both to seed the initial matches and within the

random search. By combining location and appearance, their algorithm outperforms PatchMatch. The authors point out that their final solutions are less coherent than the ones found by PatchMatch, where the coherency measure is based on neighbouring patches in the target image matched to neighbouring patches in the source image. This is a consequence to adding another type of propagation, namely the appearance-based propagation. However it is an interesting observation with respect to the original PatchMatch algorithm, which tends to find very locally coherent solutions, even if not optimal, because of the propagation step. This can be seen as a form of *implicit* smoothness, which is a byproduct of the propagation step, as mentioned previously.

[HaCohen et al., 2011] propose the Non-Rigid Dense Correspondence (NRDC) algorithm, which is based on PatchMatch and aims at recovering a dense and coherence field of matches under non-rigid transformations and different lighting conditions. It exploits the implicit coherence resulting from PatchMatch to further improve the correspondence by defining a global colour model. Two main steps are involved. The first one is an aggregation step which consists in detecting consistent regions of matches. This relies heavily on the implicit coherence of the match field computed by PatchMatch which we mention above. Then, the authors introduce a global non-linear parametric colour model which then becomes part of the matching process, as it is used to alter the colours of the patches when evaluating the similarity measure. These global parameters are also optimised, and updated after each iteration. The authors point out several limitations, such as difficulty finding reliable matches in smooth regions and difficulty in handling change of background, due to the lack of adequate support region weighting. Furthermore, only one global colour model is taken into account, which might be insufficient in certain cases. The algorithm is used for several applications, such as local colour transfer and deblurring, however it does not explicitly model any smoothness, which we can consider as another limitation. We can further hypothesise that this algorithm would not perform as well with a nearest neighbour algorithm that does not provide such initial coherence, for example CSH, as it uses the piecewise constant nature of the PatchMatch solution in order to build the colour model.

The randomised motion estimation [Boltz and Nielsen, 2010] proposes to compute optical flow between two images, using a segmentation of the image organised in a tree structure combined with PatchMatch. Naming their algorithm “GraphMatch”, the authors assume that a segmentation tree of the image is available. Then, a variation of PatchMatch is applied on super pixels rather than standard patches and finally a hierarchical propagation, instead of spatial, is used to refine the solution. This is one of the only methods that propose to compute optical flow via a PatchMatch-based algorithm, and the use of super-pixels can be seen as an attempt to introduce smoothness in the solution.

PatchMatch is also used for stereo matching in [Bleyer et al., 2011] where its efficiency makes the overparametrisation of the problem into three dimensions possible. This is discussed in section 2.5 of this literature review and further in detail in Chapter 4.

[He et al., 2011a] use PatchMatch to perform alpha matting. Given a trimap, which is a segmentation of the scene into background, foreground, and unknown pixels, their goal is to accurately model each pixel of the unknown region as a linear combination of one background and one foreground pixel.

They define a new search space where each point represents a combination of one foreground and one background pixel. To build this space they sort each foreground and background pixels according to a criterion (colour/intensity/position), the choice of which does not have a large influence on the final result. While the propagation step of PatchMatch is run spatially, the randomisation is performed in this new space. Post processing consisting in solving the matting Laplacian is applied as a final step to enforce smoothness on the result. This algorithm outputs reasonable alpha matting masks, and it illustrates another type of application where PatchMatch can be beneficial, however the only form of smoothness mechanism is the post-processing step.

In their work, [Mansfield et al., 2011] investigate different optimisation methods for image completion. The optimisation process is split into two steps: the search for nearest patches in the current target image, and the generation of a new target image from the nearest neighbour field. They consider two types of variables: discrete, such as positions in the image, and continuous, such as rotation, scale and photometric transformations. They combine PatchMatch to perform the optimisation over the discrete variables while a Levenberg-Marquadt algorithm is used to optimise on the continuous variables. This method presents an interesting approach to image completion by combining PatchMatch and a global optimiser, however no smoothness term is taken into account in the model.

Finally, [He and Sun, 2012] use a classical KD-tree search formulation, with several improvements. They first reduce the patch dimensionality through the Walsh-Hadamard Transform (WHT) [Hel-Or and Hel-Or, 2005] which is faster than running PCA and provides a 24D descriptor for each patch. They then proceed to build a traditional KD-tree using these vector using the median split scheme. The core of their improvements relies on the search strategy within the KD-tree. They devise a propagation-assisted KD-tree search where instead of performing classical back-tracking, they use the leafs of the tree returned by the search results of neighbouring pixels, which is a step inspired by PatchMatch. They also run a re-ranking operation where the best candidate patches which were found using the compressed 24D vector representation are then compared back into their original dimensional space for better accuracy. This algorithm shows improved performance: up to 10-20x speedup compared to PatchMatch for the same accuracy, and a 2-5 speedup compared to the more recent CSH. However, PatchMatch has several advantages over this method: the generalisation to multiple scales and rotations is not possible in these settings without duplicating the transformed patches into the trees. Also, PatchMatch has a greater range of similarity measures that can be used, and is also much more memory efficient as every computation is performed *on-the-fly*.

A few other methods incorporate PatchMatch into their algorithm. [Rivera and Gould, 2011] perform simultaneous multi-class labelling over image sets, where PatchMatch is used to establish correspondence between image pairs, and add edges to their conditional random field, assuming that patches matched together are likely to belong to the same object class. [Shahar et al., 2011] extend PatchMatch to 3D space-time patches to perform super-resolution, both in the spatial and temporal domain, on videos. [Klaudiny and Hilton, 2011], [Tae-o sot and Nishihara, 2012] and [He et al., 2011b] also use PatchMatch for nearest neighbour evaluation, however we have not encountered any work which explic-

itly models the smoothness within the PatchMatch framework, which is the main contribution of this thesis.

2.4 Belief Propagation

The Belief Propagation algorithm [Pearl, 1988] is a well established method used to perform inference on Bayesian networks, pairwise Markov Random Fields and more generally Factor Graphs. It is a message passing algorithm which consists in sending messages across nodes of the graph to estimate marginals. This is typically called the sum-product algorithm, but a variation exists which can estimate the Maximum A Posteriori (MAP) solution. On tree-shaped graphs, Belief Propagation gives an exact solution, while on more general graphs containing loops, a variation called Loopy Belief Propagation which is an iterative algorithm can be run to produce an estimation of the solution. Belief Propagation can be converted to an energy based formulation by turning the distributions used into log space. It is described in more detail in Chapter 3 as it is a core part of this thesis. It has been commonly used in Machine Vision [Felzenszwalb and Huttenlocher, 2006] as it is a powerful tool to minimise energies containing a smoothness term, which are often needed in vision applications. However, standard formulations of the algorithm are defined on a discrete search space, with discrete random variables. Due to the computational expense of the algorithm, it is not convenient to run it with too many possible labels, and even less so on a finely discretised continuous space. Several methods exist in the literature, attempting to address this weakness. We review them in this section.

[Minka et al., 2005] propose general guidelines for message passing algorithms using approximation in the form of three steps: first, an approximation family is chosen, and used to approximate continuous distributions. Then, a divergence measure is picked, with which it is possible to measure the quality of the approximation made with the chosen family. Finally, an optimisation algorithm is designed to find the best possible approximation.

[Isard et al., 2009] follow the above guidelines and make contributions in all steps. First, they propose to use a bounded number of piecewise-constant probability densities as their approximation family. Their algorithm also uses the KL-divergence [Minka et al., 2005] as an approximation quality measure, which leads to an intuitive algorithm seeking for a distribution of minimum entropy. The optimisation is greedy and consists in traversing axis-aligned kd-trees. The authors point out that their algorithm is suitable for spaces of low to moderate dimensions, which shows limitations when dealing with high-dimensional spaces.

In their work, [Sudderth et al., 2010] model messages with kernel density estimates. In other words, particles (samples) are used, each of them being associated with a regularising kernel. In their paper the authors show the use of mixtures of Gaussians, but point out that other choice of kernels are possible. Using multiple kernels results in a rapid increase of the number of kernels involved in message multiplication. For example, if d messages, represented by M Gaussians each, are multiplied together, a resulting mixture of M^d Gaussian is necessary to accurately represent the product, and this multiplication of terms is exponential as further messages need to be multiplied in turn. Therefore, in order to stay within the same family of approximation (only using a "budget" of M Gaussians per message), the product of mes-

sages needs to be approximated. To do so, the authors use a Gibbs sampler [Geman and Geman, 1984] to efficiently draw samples from this product and build a resulting message within the budget of Gaussians allowed. The message integration step of BP is achieved through a similar approach. A related method has been developed by [Isard, 2003], where the differences with the Nonparametric Belief propagation are that the kernel estimates are not biased and the samples drawn from the message distribution are less noisy. However the authors point out that the method of [Sudderth et al., 2010] allows more general forms of potential.

[Pal et al., 2006] propose a method that can be applied to large discrete spaces. They maintain sparse local marginals by using Kronecker delta functions. Those delta functions are selectively pruned in a way that maximises a KL-divergence measure. This can be seen as a form of compression of the messages, where the least informative parts of the messages are discarded. The practice of ignoring state configurations with low marginal is called *beam search*. The authors introduce a variation of the beam search process which can be applied to sum-product inference by considering the KL-divergence to compress marginal distributions.

The Particle Belief Propagation [Ihler et al., 2009] method proposes another type of approximation using particles (or samples) which do not use regularising kernels, contrary to [Sudderth et al., 2010]. The algorithm samples a set of particles at each node, drawn from a certain sampling distribution. The authors point out the key difference, and advantage compared to Nonparametric Belief Propagation, which is that while in NBP a sending node s draws new samples to represent the outgoing message destined to t, in PBP the message going to t is represented with the sample set of t directly. This has the benefit that message products are well defined, as all incoming messages at a node are already expressed in the set of particles of this node, and thus there is no need for using a regularising kernel. An important consideration is the sampling distributions used to draw particles. While sampling from the marginal distributions would be ideal, it is not possible, as these are not known, and are the outcome of the iterative optimisation scheme. However, the belief estimate can be easily evaluated at any point, and therefore a short Markov Chain Monte Carlo (MCMC) simulation can be used to draw new samples. The potentials can have any form, as only point-wise evaluations are required, which is an advantage compared to being restricted to Gaussians for example. A variation of this method called the Max-Product Particle Belief Propagation (MP-PBP) has been proposed by [Kothapa et al., 2011], where the algorithm is adapted to the computation of the MAP solution rather than the marginal. This algorithm is described more in detail in section 3. Our work is based on this particular variation of Belief Propagation, as it shares similarities with the PatchMatch algorithm which is explained in Chapter 3, and we aim at improving it. Finally [Ihler et al., 2009] extend PBP to the Tree-Reweighted Belief Propagation message passing algorithm [Wainwright et al., 2005].

[Felzenszwalb and Huttenlocher, 2006] propose a few acceleration steps to improve the performance of Belief Propagation, applied to vision applications. They point out that with certain cost functions, such as the Potts model, the minimisation performed in the message computation can be expressed in a form that reduces the computation complexity. For linear cost functions, the minimisation can be

seen as a lower envelope computation which is in this case related to a distance transform calculation, and a similar approach is used for quadratic functions. For bipartite graphs, they perform the message updates by computing messages to each region of the graph in an alternate manner, reducing the computation cost in half. Finally, another mechanism used in this method is a coarse-to-fine version of Belief Propagation, which allows for long range interactions at coarse levels.

[Noorshams and Wainwright, 2011] use a randomised approach to BP in order to perform message update. The authors reduce the complexity of the algorithm by passing partial -and randomly chosen- information form node to node. In other words, messages are incrementally updated in a stochastic way. Applied to stereo matching it proves to be 3-4 times faster per iteration, but requires more iterations to reach the same quality as BP, which is expected as it needs more trials to fully propagate the information between nodes.

Finally, [Komodakis, 2006] designed a method to perform image completion using Belief Propagation. However the authors point out that the high number of possible labels for each pixel is a problem with respect to the complexity of the computation, which is in general an acknowledged difficulty when using Belief Propagation algorithms. To remedy to this problem the authors propose a Priority Belief Propagation algorithm which has two main components. First, the message scheduling is priority-based. In other words, the nodes that are the most confidents about their labels are the first ones to send messages, which accelerates the convergence of the algorithm. A priority list is used to refer to nodes and they are updated in order within this list. Then, they also implement label pruning, labels that are unlikely to be assigned to a node are discarded. The remaining labels are the "active labels", and this scheme further improves the performance.

Most of these approaches attempt at making Belief Propagation applicable to continuous, or large discrete spaces. However, due to the high computational costs involved, this still remains an open research problem, especially when working with correspondence fields between images, where the search space is often very large.

2.5 Stereo Matching

In this section we review the various stereo matching techniques present in the literature. [Scharstein and Szeliski, 2002] propose an algorithmic taxonomy that breaks down the typical stereo matching algorithm into several building blocks:

- matching cost computation
- cost aggregation, where the matching costs over a support region are combined
- disparity computation (local and global methods)
- disparity refinement

While these steps are not always followed by all algorithms in this particular order, they are often present in some form. The disparity computation step can be approached by two families of techniques: local and global methods. Local methods (also known as window-based), base their computation on the similarity

of intensities in a finite window around a given pixel, while global methods establish an energy-based framework and aim at minimising the proposed energy, often combining unary and pairwise terms for smoothness, using various optimisation techniques. Other algorithms do not explicitly state an energy term to optimise, but behave in a similar way, as we will see below. In this section we update and complete this taxonomy with the most recent stereo matching techniques.

2.5.1 Cost Computation

The similarity measures presented in section 2.1 are widely used in stereo matching. The most popular which have been used for decades are SAD [Kanade et al., 1995], and SSD [Hannah, 1974]. Alternatively, gradient-based measures are also commonly used to provide robustness to change in illumination [Scharstein, 1994]. Some techniques combine both colour and gradient information [Rhemann et al., 2011, Bleyer et al., 2011] while others perform a pre-processing step on the images prior to the similarity measure computation [Stefano et al., 2004]. NCC is also found in the literature as a common alternative to SSD [Psarakis and Evangelidis, 2005], as well as mutual information [Hosni et al., 2009]. Finally, truncating the error measure is common practice to provide robustness to outliers [Rhemann et al., 2011, Bleyer et al., 2011].

2.5.2 Cost Aggregation

Methods using patches in order to calculate the cost of a possible match use various schemes to aggregate the cost of all pixels of the supporting area. Some techniques, such as [Mühlmann et al., 2002, Stefano et al., 2004] simply sum the costs of all the pixels in the window, which implicitly assumes a fronto-parrallel organisation of the scene and does not take into account edges. The guided filter [He et al., 2010], the bilateral filter [Tomasi and Manduchi, 1998] and the Adaptive Support Weight method [Yoon and Kweon, 2006, Bleyer et al., 2011] are often used during aggregation to provide edge-awareness. A variation of the Adaptive Support Weight is the Geodesic Support Weights used by [Hosni et al., 2009], where pixels with high weights must not only have the same appearance as the middle pixel of the window, but also have a path connected to the middle pixel which does not vary significantly in colour. [Min et al., 2011] use histograms to perform aggregation, together with weights similar to the bilateral filter, to count each pixel's vote towards the final solution. A multi-scale aggregation approach has been devised by [Min and Sohn, 2008] for further acceleration and reliability.

2.5.3 Disparity Computation - Local Methods

Local methods estimate the cost of a candidate disparity by computing intensity errors in a window around each pixel. Furthermore, the disparities are assigned following the winnner-take-all (WTA) scheme, which states that the candidate with the lowest error is assigned to the final match. This implies that no neighbouring or global information is taken into account, except from the aggregation step.

In their work, [Rhemann et al., 2011] present an algorithm that performs remarkably well on the Middlebury dataset, while running in real-time. The authors build a cost volume, also known as Disparity Image Space (DIS). After this, they aggregate the costs using a support window, which is determined from the guided filter. This provides a way of regularising the solution while preserving edges. Similarly,

[Mizukami et al., 2012] build a cost volume and use an interpolation method to provide costs at sub-pixel positions. Then, they filter the cost volume using a guided filter [He et al., 2010] and finally the disparity at each pixel is determined by using a local coarse-to-fine scheme. An advantage of this method is that it can be easily parallelised. Other methods, such as [Richardt et al., 2010] use a fast bilateral filter for real time performances, but are limited to grey scale images due to heavy memory requirements.

[Bleyer et al., 2011] propose an algorithm based on PatchMatch applied to the stereo matching problem. The authors aim at estimating a supporting plane at each pixel, in order to explain the existing variation of the disparity values on slanted surfaces in a natural way. By over-parametrising the disparity computation problem to a plane estimation problem, they propose a robust matching cost which produces state-of-the-art results at subpixel accuracy levels. However, while the algorithm benefits from an implicit smoothness due to the propagation mechanism of PatchMatch, it also suffers from the lack of explicit smoothness in the optimisation. We describe this algorithm in more detail in Chapter 4 and use our new optimisation technique to address some of its weaknesses.

2.5.4 Disparity Computation - Global Methods

Global methods often use an energy minimisation framework, where the sought solution is the configuration of disparities that minimises an energy which typically has the following form:

$$E(u) = E_d(u) + \lambda E_s(u). \tag{2.10}$$

where u is a possible solution, E_d is the data term (also known as unary term), which indicates how well the solution agrees with the data present in both images, E_s is a smoothness term which encourages neighbouring pixels to have similar disparity (or similar parameters in the case of an alternative parametrisation), and λ controls the influence of the smoothness term over the data term. The smoothness term is often a pairwise term, but higher order methods also exist in the literature [Woodford et al., 2009]. There are various optimisers that can be used to minimise this energy.

Several methods use a Graph-Cut algorithm [Boykov and Jolly, 2001], such as [Kolmogorov and Zabih, 2001] and [Wei and Quan, 2005], where an α-expansion scheme is used to handle more than two labels for the disparity. A drawback of using graph cuts is that the solution is often limited to integer disparities.

Another family of algorithms relies on the Belief Propagation algorithm [Pearl, 1988] to carry out the optimisation. In their work, [Sun et al., 2003] formulate the stereo matching problem using Markov Random Fields (MRFs) and estimate the optimal solution using Belief Propagation. [Klaus et al., 2006] also use Belief Propagation as the optimisation method, but aim at estimating a disparity plane at each pixel, in a similar fashion as [Bleyer et al., 2011]. Several Belief Propagation methods aim at improving the computational time for the stereo matching application, such as [Yang et al., 2009] who use a hierarchical approach, and [Felzenszwalb and Huttenlocher, 2006] who efficiently compute messages by considering the relative difference between labels, use bipartite graphs and a multiscale variant of Belief Propagation for optimisation. [Tappen and Freeman, 2003] provide a comparison between Graph-Cut and Belief propagation by applying both to solve the same MRF, using a Potts model

[Boykov et al., 2001], and conclude that they are comparable: while Graph-Cuts provide slightly lower energy solutions, this does not always correlate with a better quality solution when evaluating the final error. However the main weakness of Belief Propagation is that computations are expensive, and optimising in a continuous space is not straightforward.

Finally, MRFs can also be solved by other techniques such as simulated annealing [Geman and Geman, 1984, Barnard, 1986], and variational approaches [Horn and Brooks, 1986, Slesareva et al., 2005], which are very popular in the Optical Flow Estimation field, as we will see below.

2.5.5 Disparity Refinement

Post processing of the disparity field is often used to improve the quality of the solution at relatively low computational cost. For methods that match the images in a bidirectional way, a useful step is the right/left consistency check, which can detect erroneously matched pixels as well as pixel residing in occluded or disoccluded areas. Additional post-processing can then be applied on these invalid pixels, such as inpainting [Bleyer et al., 2011]. The median filter is also commonly used to remove unwanted matches [Birchfield and Tomasi, 1999]. Finally, it is common for algorithms that evaluate disparities at integer positions to perform refinement in the form of curve fitting allowing a final sub-pixel solution [Stefano et al., 2004].

2.6 Optical Flow Estimation

Optical Flow is defined as the apparent motion of the pixels in the image. A related term that is often used is the "motion field", which is the 3D motion of the objects, whose 2D projection on the image space is the optical flow. [Baker et al., 2011] created a taxonomy for Optical Flow algorithm, similar to that of [Scharstein and Szeliski, 2002] for the stereo matching methods. In this section we present an overview of the different optical flow algorithms.

In the same fashion as the global methods of stereo algorithms, most optical flow techniques aim at solving an energy minimisation problem, where the energy is defined per Equation 2.10. Each algorithm using this type of energy can be broken down into two components: the model and the optimiser. The model can be further separated into two sub-components which are (1) the choice of the data term E_d, and (2) the choice of the smoothness term (often called prior) E_s.

2.6.1 Models

Most models follow the classical formulation of [Horn and Schunck, 1981], which establishes the constancy of the image in some way (e.g. brightness constancy), and also models the expected variation of the flow across the image (e.g. flow smoothness). These two constraints form the data and the prior (or smoothness) terms.

The data term, often a brightness constancy assumption, is a key element of the model. The original formulation of [Horn and Schunck, 1981] using the L2 norm for the data term is sensitive to outliers, especially near occlusion boundaries where pixels might disappear. [Black and Anandan, 1996] follow the same guidelines but use a robust Lorentzian penalty function, which has also been part of more recent

algorithms [Brox et al., 2004, Wedel et al., 2009]. Other robust penalty functions have been proposed, such as the Geman-McClure in [Lempitsky et al., 2008]. However it is not clear which penalty function performs best: it is possible that none is best for all data, and that it can be beneficial to use different algorithms on different parts of the image, which is the approach used by [Mac Aodha et al., 2010]. Note that the brightness constancy assumption can be applied on elements other than raw colour values of the image. [Brox et al., 2004] use gradient values, which are often more robust to change in illumination, while [Liu et al., 2011] use SIFT descriptors which are also invariant to rotation and scale. Change in illumination can also be explicitly modelled as well as blur, such as in the work carried out by [Seitz and Baker, 2009], however this increases the ambiguity of the data term.

The smoothness term, or prior, helps in reducing the level of ambiguity brought by the data term and regularises the solution. The most common priors are first order, meaning that they are essentially a pairwise term and thus act on the gradient of the flow field. Again the choice of the penalty function for the prior is crucial to the quality of the algorithm. [Horn and Schunck, 1981], as mentioned before, propose a quadratic term for the prior, yielding Tikhonov regularisation [Tikhonov, 1963]. This work is considered as one of the milestones of optical flow algorithms, and a significant amount of research has been built upon it. The total variation (TV) method typically uses a L1 formulation for the penalty function of the flow field gradient to regularise the solution [Brox et al., 2004, Wedel et al., 2009] , and belong to the class of variational optimisers, described in the next section. An important property of the regularisation term is edge preservation: [Werlberger et al., 2009] introduce an anisotropic, image based regularisation term replacing the widely used isotropic TV one; [Sun et al., 2008] model the smoothness term using a Steerable Random Field, where the derivatives of the flow are steered by the image structure; [Lee et al., 2010] use a bilateral filter within the regularisation term. Segmentation information can be incorporated into the model, such as in [Xu et al., 2008]. Learning is also a valid option: [Roth and Black, 2007] learn the prior model to regularise the flow using a Field-of-Expert formulation. Higher order regularisation terms can be used to penalise differences of the second-order derivatives [Trobin et al., 2008]. Over-parameterising the flow has the same effect: for example [Nir et al., 2008] use an affine prior to allow for slanted surfaces. Finally rigidity priors, consisting in finding a solution agreeing with the epipolar geometry have been proposed [Nir et al., 2008], [Wedel et al., 2009]. This is related to the method we devise in chapter 6.

2.6.2 Optimisers

A first family of optimisers used to minimise an energy is based on a gradient descent scheme. The most simple algorithm of this kind is the steepest decent algorithm seen in [Baker and Matthews, 2004]. Other known methods are the Newton method, Gauss-Newton method, Quasi-Newton method and the Levenberg-Marquardt algorithm [More, 1978].

A second type of methods are the variational algorithms where the flow values are treated as 2D functions u and v. [Sun et al., 2010a] point out that the classical Horn and Schunck model works surprisingly well with modern optimisers, and the energy function that the authors define is typically minimised

using a variational algorithm. The global energy function is written in the following form:

$$E_{\text{global}} = \iint E(u(x,y), v(x,y), x, y, u_x, u_y, v_x, v_y)dxdy, \tag{2.11}$$

where $u_x = \frac{\partial u}{\partial x}$, $u_y = \frac{\partial u}{\partial y}$, $v_x = \frac{\partial v}{\partial x}$ and $v_y = \frac{\partial v}{\partial y}$.

Minimising E_{Global} leads to the following Euler-Lagrange equations:

$$\frac{\partial E_{\text{global}}}{\partial u} - \frac{\partial}{\partial x}\frac{\partial E_{global}}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial E_{global}}{\partial u_y}, \tag{2.12}$$

$$\frac{\partial E_{\text{global}}}{\partial v} - \frac{\partial}{\partial x}\frac{\partial E_{global}}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial E_{global}}{\partial v_y}. \tag{2.13}$$

These equations can be solved in many different ways, such as using the Jacobi method [Bathe and Wilson, 1976], the Gauss-Seidel method, Successive Over-Relaxation [Young, 2003] and the Conjugate Gradient algorithm [Fletcher, 1976, Sun et al., 2010b].

A class of variational algorithms that are particularly popular in optical flow computations is the total variation regulariser, which is coupled with the robust L1 norm for the data term, capable of preserving discontinuities in the flow field: for example [Zach et al., 2007] approximate the L1 norm with the Charbonnier penalty function which is a differentiable approximation in order to reduce the computational difficulties, and use a dual formulation of the energy, which is then optimised using an alternative scheme on the GPU.

Recently, [Xu et al., 2012b] use the TV formulation combined with an initialisation using SIFT features to recover large motion more effectively, and obtain good performance. Other variational methods also exist such as the scale invariant flow of [Xu et al., 2012a] which is able to establish correspondence across different scales, the non-local TV of [Werlberger et al., 2010], or the method of [Brox et al., 2009] which is adapted to large displacements.

There are several additional steps used by optical flow algorithms that considered common practice. The first is to use a coarse-to-fine strategy, which involves first computing the flow at a subsampled resolution, to then provide an estimation of the solution for the next levels of the pyramid. Each estimated flow field can be used to warp one image towards the other one, thus leaving an incremental flow to compute in the next iterations [Bruhn et al., 2005]. This is particularly helpful when the flow field contains large displacements.

Graduated Non-Convexity [Black and Anandan, 1996] is also commonly used to provide a convex estimation of the objective function, which can then be more easily minimised.

Occlusion reasoning, depth ordering and explicit layer representation help in having a better understanding of the scene and separating the estimation of motion boundaries from the smooth estimation of the flow, such as in the work carried out by [Sun et al., 2010b] which performs particularly well on the Middlebury dataset.

Finally, several filtering operations are also typically used, firstly as pre-processing, such as the Rudin-Osher-Fatemi (ROF) technique [Rudin et al., 1992] which aims at reducing the influence of the illumination change between the two images. Additionally filtering can be used as a post-processing

step to improve the final flow, for example with the use of a median filter, which, according to [Sun et al., 2010a], is one of the most influential steps.

Very few methods used the PatchMatch algorithm for optical flow computation [Boltz and Nielsen, 2010], and this could be due to the lack of explicit smoothness in the optimisation. However, we will see in Chapter 5 that is is a valid possibility, and we further improve it by adding this missing explicit smoothness to the method via our algorithm described in Chapter 3.

2.7 Our Approach

In this work we aim at addressing the weaknesses of PatchMatch and Belief Propagation in order to perform fast correspondence field estimation on continuous spaces while keeping the possibility of optimising energies containing pairwise terms. We study the similarities between these two algorithms and express them under the same framework. This allows us to create a new algorithm, that we call PatchMatch Belief Propagation (PMBP), which combines the strength of both PatchMatch and Belief Propagation. This new optimiser can be seen a generalisation of PatchMatch expressed in a Particle Belief Propagation framework and is controlled by a smoothness weight. This means that it can reproduce the exact same outputs of PatchMatch under certain settings, while being also able to introduce smoothness in the solution by using the mechanisms of Belief Propagation while retaining similar computational time to PatchMatch. We show the benefits of this new optimiser, and use it within different applications. First, we extend the work of [Bleyer et al., 2011] and create a new stereo matching algorithm which estimates supporting disparity planes at each pixels and introduce a new pairwise term in order to benefit from the smoothness given by PMBP. We show state-of-the-art outputs on the popular Middlebury stereo benchmark [Scharstein and Szeliski, 2002]. We then apply our algorithm to optical flow computations. We first motivate this work by calculating optical flow and performing fast video editing with PatchMatch, which has not been done before. We then devise several models that include pairwise terms and can be optimised with PMBP. These include a 5D model using patch translation, rotation and anisotropic scaling, a 6D affine model that can capture more complex deformation, and finally a 9D model using homographies that are generated via a geometrically based model. We show that this model can naturally handle the smoothness variation of the flow on slanted surface and produce competitive results especially on large displacement cases.

Chapter 3

PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation

Our work draws a new connection between two existing algorithms for estimation of correspondence fields between images: Belief Propagation [Pearl, 1988][Yedidia et al., 2005] and PatchMatch [Barnes et al., 2009] [Barnes et al., 2010]. Correspondence fields arise in problems such as dense stereo reconstruction, optical flow estimation, and a variety of computational photography applications such as recolouring, deblurring, high dynamic range imaging, and inpainting. By analysing the connection between the two methods, we obtain a new algorithm which has performance superior to both its antecedents. The first contribution of our work is a formulation of PatchMatch and Belief Propagation in terms that allow the connection between the two to be clearly described. Our second contribution is in the use of this analysis to define a new algorithm: PatchMatch Belief Propagation (PMBP) [Besse et al., 2012, Besse et al., 2013b] which is more accurate than PatchMatch and orders of magnitude faster than Belief Propagation.

3.1 Energy Minimisation and Correspondence Fields

We consider the correspondence field estimation task as an energy minimisation problem. We parametrize the correspondence field by a vector grid $\mathbf{u} = \{\mathbf{u}_s\}_{s=1}^{n}$ where s indexes *nodes*, typically corresponding to image patches, and $\mathbf{u}_s \in \mathbb{R}$ parametrizes the correspondence vector at node s. We denote the output domain of $\mathbf{u}_s$ as $\mathcal{U}_s$ and the overall output domain of a configuration $\mathbf{u}$ as $\mathcal{U}$. We shall consider a special type of energy, combining *unary* and *pairwise* energies

$$E(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \sum_{s=1}^{n} \psi_s(\mathbf{u}_s) + \lambda \sum_{s=1}^{n} \left[\sum_{t \in N(s)} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) \right]. \tag{3.1}$$

with $N(s)$ being the set of *pairwise neighbours* of node s. The unary energy ψ_s computes the local evidence for the correspondence $\mathbf{u}_s$ and is often called the *data term*. The pairwise energy ψ_{st} encourages smoothness between neighbouring correspondence vectors $\mathbf{u}_s$ and $\mathbf{u}_t$. Note that this energy formulation can be converted to a probabilistic form using the convention:

$$p(\mathbf{u}_1, \ldots, \mathbf{u}_n) = \exp(-E(\mathbf{u}_1, \ldots, \mathbf{u}_n)). \tag{3.2}$$

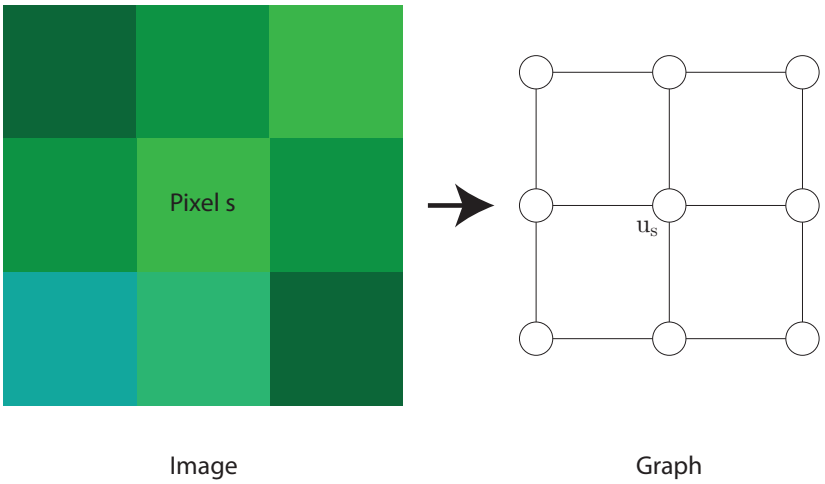


Figure 3.1: Parametrization of the correspondence field. Each pixel corresponds to a node in a graph, and the random variable u_s corresponds to the possible correspondence states of pixel s.

In order to perform comparisons between the quality and performance of different algorithms throughout this chapter, we consider a simple 2D patch-based correspondence field estimation task between two images using the energies defined below. We refer to this as the "2D Patch Correspondence Estimation" (2D-PCE) and will use it later on in this chapter.

Let $\mathbf{u}_s = (u_s, v_s)$ be the parametrisation of a 2D flow field between images I_1 and I_2. This means that for each pixel s in I_1, u_s is a 2D random variable that represents the correspondence of s into I_2 as illustrated in figure 3.2. We define a weighted patch data term where (x_s, y_s) are the image coordinates of pixel s as

$$\psi_s^{\text{wpf}}\left(\begin{bmatrix} u_s \\ v_s \end{bmatrix}\right) = \sum_{i=-h}^{h} \sum_{j=-h}^{h} w_{sij} \Big\| I_1\big(x_s + i,\ y_s + j\big) - I_2\big(x_s + i + u_s,\ y_s + j + v_s\big) \Big\|. \tag{3.3}$$

where the parameters of the state are displayed in red. Here, the weights w_{sij} are precomputed based on the intensity values surrounding pixel s, and the norm $\|\cdot\|$ represents magnitude of difference in an appropriate colour space. For 2D-PCE we use $w_{sij} = 1$. For a stereo correspondence, which is a correspondence along the same scanline, with $\mathbf{u}_s = [\Delta_s]$ being the single scalar disparity, the equivalent data term is $\psi_s^{\text{wps}}([\Delta_s]) = \psi_s^{\text{wpf}}([\Delta_s, 0]^\top)$. This is a simple model which assumes a constant correspondence field in the $(2h+1) \times (2h+1)$ patch surrounding every pixel. With large h, this oversmoothes the solution, even with clever choices of w_{sij}, which can be improved by using more complex parameterisations of the correspondence field, as discussed in Chapter 4 for stereo matching and Chapter 5 for optical flow computations. Furthermore, large h yields expensive computation due to the systematic comparison between large patches. On the other hand, smaller h make the algorithm more tractable, but increase the ambiguity of the data term computation, as patches are less distinctive. Introducing pairwise terms encouraging smoothness on the correspondence field decreases this ambiguity, which is why we use an energy of the form seen in equation 3.1 in our work.

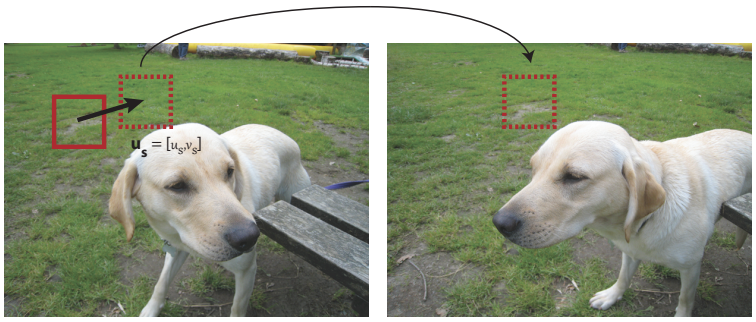


Figure 3.2: Correspondence field estimation. For each patch of the target image we are looking for a patch in the source image such that the global energy is minimised. The correspondence field is parameterised with states representing the offset of the matches with respect to the position of the original pixel in the target image. Each pixel of the target image is associated with a node in the graph. Each node can be seen as a random variable. A state is a possible value for a random variable.

We also define a simple pairwise term using the same 2D parametrisation:

$$\psi_{st}^{11}(\begin{bmatrix} u_s \\ v_s \end{bmatrix}, \begin{bmatrix} u_t \\ v_t \end{bmatrix}) = \min(|u_s - u_t| + |v_s - v_t|, \tau) \tag{3.4}$$

where the parameters of the states are displayed in red, and where τ is a truncation term. It is essentially a $L1$ norm between two states, which cannot take values greater than τ, in order to add robustness [Bleyer et al., 2011]. This pairwise term has the effect of encouraging neighbouring correspondence vectors to be equal to each other.

3.2 Belief Propagation

Belief Propagation (BP) is a *message passing* algorithm used to solve inference problems and minimise energies such as equation 3.1. There are two main variants of the algorithm. The first one is called Max-Product Belief Propagation, which aims at computing the MAP solution $\bar{\mathbf{u}}$ defined as

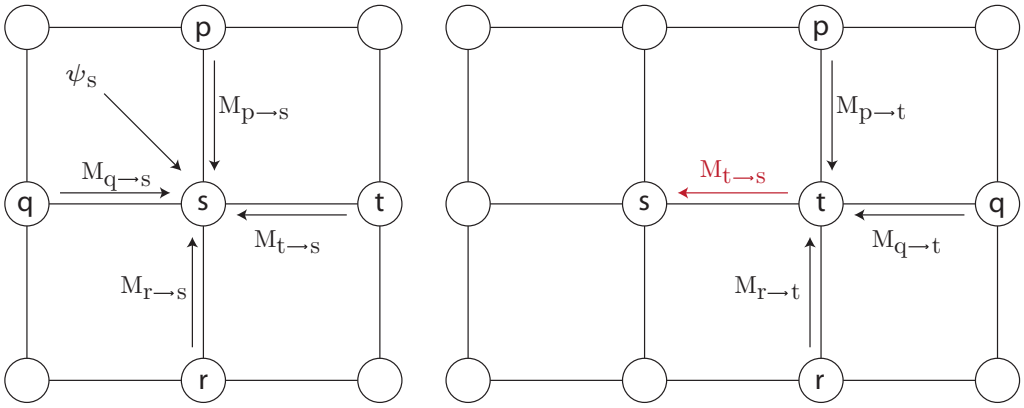
$$\bar{\mathbf{u}} = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmax}}\, p(\mathbf{u}) = \underset{\mathbf{u} \in \mathcal{U}}{\operatorname{argmin}}\, E(\mathbf{u}). \tag{3.5}$$

The second version is called Sum-Product Belief Propagation, and is used to compute the maximum values of the marginal distributions:

$$\mathbf{u}^* = \left\{ \underset{\mathbf{u}_s \in \mathcal{U}_s}{\operatorname{argmax}}\, p(\mathbf{u}_s) \right\}_{s=1}^{N}, \tag{3.6}$$

where N is the number of nodes in the graph.

In this work we focus on retrieving the MAP solution, but it is worth mentioning that both versions of the BP algorithm have similarities. In tree-shaped graphs, BP gives an exact solution after traversing the graph twice. In general graphs, BP can still be run in an iterative fashion and is called Loopy BP, but is not guaranteed to converge [Yedidia et al., 2005]. To fully characterise BP, we need to define what we call *beliefs* and *messages* which are parts of the core mechanisms of the algorithm.



(a) Belief computation (b) Message computation

Figure 3.3: Belief and message computation. (a) The belief computation at node s is the product (or sum in log-space) of all the incoming messages with the unary term; (b) The message computation from node t to node s involves a maximisation (or minimisation in log-space) over the states of t of the product (sum in log-space) of all messages incoming at t (except from $M_{s\to t}$) multiplied with (added to in log-space) the pairwise term.

3.2.1 Beliefs

The BP algorithm is an iterative algorithm that builds estimates of what is called the *belief* at each node. A belief indicates how likely it is for each state to be the solution at the node. After convergence of the algorithm, the MAP solution can be estimated by retrieving the max-belief state at each node. We recall that probabilistic form can be converted to an energy-based formulation, using the convention $E = -log(p)$ where p defines a probability and E defines the corresponding energy as mentioned before. In our work, we chose to represent the problem with the energy-based formulation, and will therefore often refer to the *disbelief*, defined as

$$B_s(\mathbf{u}_s) = -\log b_s(\mathbf{u}_s), \tag{3.7}$$

where b_s is the belief at node s . In this context, the MAP solution is estimated by retrieving the min-disbelief states. Formally, the disbelief at node s is computed as:

$$B_s(\mathbf{u}_s) := \psi_s(\mathbf{u}_s) \; + \sum_{t \in N(s)} M_{t\to s}(\mathbf{u}_s), \tag{3.8}$$

where ψ_s is the data term at node s, and $M_{t\to s}(\mathbf{u}_s)$ is the message from node s to node t, which we define below. The data term is also called the unary term, and evaluate how well a states agrees with the available data. For example, in the 2D-PCE task that we defined before, the data term is ψ_s^{wpf} and encourages the correspondence to be similar in colour to the patch located around pixel s (brightness constancy assumption). As illustrated on figure 3.3a, the disbelief is computed by gathering information about all the incoming messages at a node.

3.2.2 Messages

In order to build the disbeliefs, as seen in equation 3.8, the algorithm uses *messages* which circulate between nodes according to a schedule ϕ. A message always has a *sender* node and a *receiver* node. The message from node t to node s represents, in words, "node t's opinions of the [negative log of the] likelihood that node s has value $\mathbf{u}_s$". This is illustrated in figure 3.3b. It is formally defined as:

$$M_{t\to s}(\mathbf{u}_s) := \min_{\mathbf{u}_t} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) + B_t(\mathbf{u}_t) - M_{s\to t}(\mathbf{u}_t). \tag{3.9}$$

where ψ_{st} is a pairwise-term. For convenience, the variable that is being minimised over is displayed in green. As messages are clearly defined in a recursive fashion, the messages on the right-hand side of equation 3.9 are those computed in the previous iteration. Further algorithmic and implementation details are discussed in section 3.6.

3.2.3 State space and limitations

We have defined until now the BP equations without specific information about the state space $\mathcal{U}_s$ for a given node s. This standard BP formulation is generally used with discrete state spaces. In order to use this formulation on a continuous state space, one would need to keep a continuous representation of beliefs and messages, but this is not possible in practice. Instead, a fine discretisation of the space could be used. However, this causes the algorithm to become quickly computationally intractable. This is a consequence of the minimisation performed in equation 3.9 which has a quadratic complexity. To exemplify this on a task involving images, we run a standard BP on 2D-PCE using two small images (50×50) and a 2D state space discretised over n values for each dimensions, yielding a total of n^2 possible states. The resulting computation time over 3 iterations with respect to the state space's size is shown in figure 3.4. We see that even for small images, the algorithm is far from performing in real time due to its complexity, and that is with relatively small values of n. For real-world applications, optical flow computations need to be performed on an almost continuous state space (floating point values) and larger images (several Mpx). To achieve this with discrete BP, the grid on which the space is discretised would need to contain tens of thousands points, therefore it is clear that it is not suitable for such applications.

As discussed in Chapter 2, several methods exist in the literature trying to address the problem of running BP on a continuous space. The key to implementing BP for continuous state variables $\mathbf{u}$ is in the representation chosen for the messages and beliefs. [Isard et al., 2009] propose a solution by discretising the space in a way that minimises a Kullback-Leibler (KL) divergence measure. [Noorshams and Wainwright, 2011] work on large discrete spaces, and use a randomisation step to incrementally and stochastically update partial messages, reducing the complexity from quadratic to linear. [Pal et al., 2006] also operate on large discrete spaces, and maintain sparse local marginals by using Kronecker delta functions, keeping only labels carrying the highest probability mass. [Sudderth et al., 2010] extend particle filters to Loopy BP, and use a regularisation kernel to ensure that message products are well defined. Finally, Particle Belief Propagation [Ihler et al., 2009, Kothapa et al., 2011], uses a simplified representation of the continuous messages and beliefs using particles. We present this algorithm in

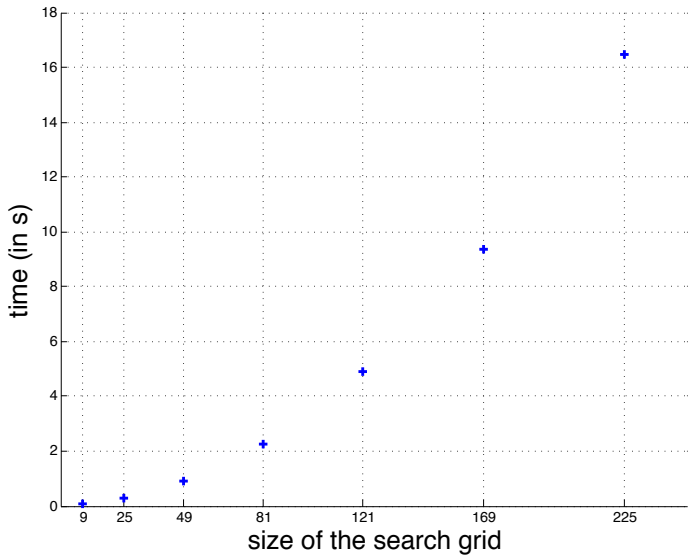


Figure 3.4: Computation time for 3 iterations of discretised BP with varying discretisation level of the state space.

detail in the next section as our work is based on it.

3.3 Particle Belief Propagation

The Particle Belief Propagation (PBP) [Ihler et al., 2009] algorithm represents continuous beliefs and messages using a finite set of particles, where a particle is a weighted sample of a distribution, and is associated with a point in the state space. Each node s is associated with a set $P_s = \{u_s^1, ..., u_s^m\}$ containing m particles. The key mechanism is that these sets of particles are not fixed, and are allowed to *evolve* in order to keep the best possible representation of both beliefs and messages. In other words, PBP consists in two separate parts: a standard BP message passing algorithm adapted to particle-based representations of beliefs and messages, and a particle resampling algorithm seeking to represent those beliefs and messages in the best possible way with the given fixed number of particles allowed. In this section we describe these two components.

3.3.1 Message Passing

The message passing mechanisms of PBP are very similar to that of normal BP, the only difference being that the min operation performed during the message computation is done only on the set of particles of the sender nodes (when computing the message $M_{t\to s}$, we call t the sender node and s the destination node). This has the effect of reducing the computational cost of the algorithm, which is now tied to the number of particles in each set, rather than the dicretisation level of the state space. The message from node t to s can now be written as

$$M_{t\to s}(\mathbf{u}_s) := \min_{\mathbf{u}_t \in P_t} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) + B_t(\mathbf{u}_t) - M_{s\to t}(\mathbf{u}_t). \tag{3.10}$$

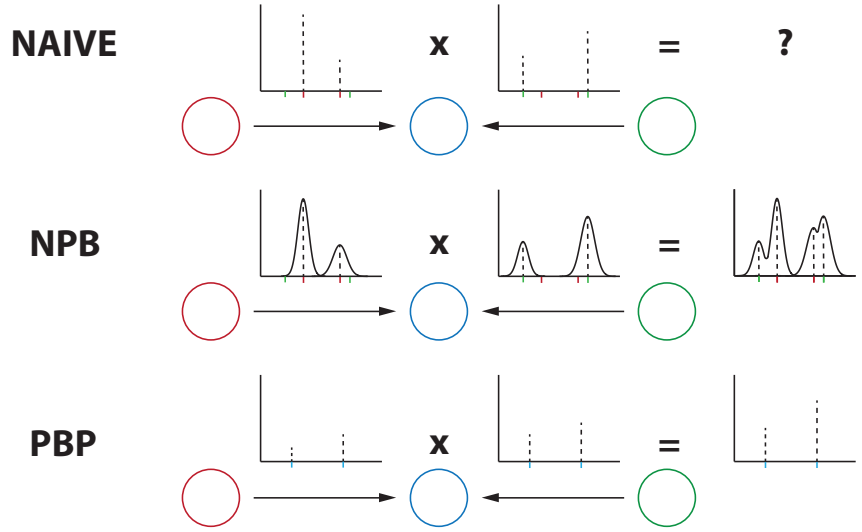


Figure 3.5: Message multiplication (probabilistic form). To calculate the belief at the blue node, we need to multiply the messages incoming from the red and the green node. A naive algorithm that sends messages represented as particles located at the states of the sender (red ticks for the red node, and green ticks for the green node) would have a fundamental problem when trying to multiply the different values together as the product would not be defined. NPB fits a Gaussian around each particle so that the message product is defined. Note that it produces a mixture of Gaussians with a higher number than the original number of Gaussians in the incoming messages. If the "representation budget" for the messages is set to n Gaussians (2 in the example), the resulting product (which contains now 4 Gaussians) needs to be represented itself by n Gaussians, and therefore a further approximation is necessary (in the example, the resulting messages contains 4 Gaussians, but needs to be approximated with 2 Gaussians). PBP ensures that the incoming messages are already expressed in the particle set of the receiver (blue node, blue ticks) which leads to a straightforward message multiplication.

where P_t is the set of particles at node t. It is worth noting is that while the minimisation is performed on the particles of t, the resulting message is a function of $\mathbf{u}_s$ and therefore is evaluated at the particle positions of s when calculating the disbelief of s. Keeping a set of particles at each node and evaluating all the incoming messages in this set of particles is an important mechanism, that can be seen as a type of "conversion" between the information outgoing from node t which is expressed in terms of particles in P_t, through the pairwise term into information about s arriving at node s in terms of particles in P_s. As a result, the evaluation of the belief at node s can still be represented by equation 3.8, as each incoming message is defined at each particle of s. In other methods, such as Nonparametric Belief Propagation (NBP) [Sudderth et al., 2002], the incoming messages do not share the same collection of samples, which requires smoothing in order to have product (in the probabilistic version), or sum (in the energy-based framework) defined everywhere. These mechanisms are illustrated in figure 3.5. Although these two algorithms might seem very different, they are actually working towards the same goal: approximating in the best possible way the output of a message computation, with a given budget of some elements used to represent messages, such as particles for PBP, or Gaussians for NBP. The particle approach has the advantage of working easily with any form of pairwise term, as it only needs to be evaluated in certain points. However, it requires an extra step which is particle resampling.

3.3.2 Particle Resampling

In PBP, the message passing step alternates with a particle resampling step, which is used to refine the current set of particles at each node in order to have a better estimation of the disbelief. [Koller et al., 1999] propose the use of the sum-product marginal at each node as sampling distribution, which is not available in our case. Another alternative would be to use the belief at each node. However this is not straightforward as the belief is an unknown distribution. On the other hand, the belief is easily evaluated, and therefore the method of choice used here is a short Markov Chain Monte Carlo (MCMC) simulation in order to obtain samples. In particular, a Metropolis-Hastings algorithm is run, using Gaussian distributions centred at each particle location to perform the random walk, and an arbitrary bandwidth parameter σ chosen so that the acceptance rate of the Metropolis-Hastings algorithm is neither too high or too low. The main advantage of PBP over discretised BP is that instead of having a fixed grid, which could be seen as fixed particles regularly sampled over the search space, it uses dynamic particles which are allowed to evolve, and the MCMC step is used to update these particles.

3.3.3 Limitations

While PBP allows continuous space optimisation, it is still not suitable for very high dimensional and/or large spaces as the convergence rate is slow. The reason behind this is that in order to correctly approximate the real messages, a large number of particles is needed. If the number of particles is too low, it is likely that the algorithm will fail to accurately compute an approximate message due to what we call "pairwise misses". A pairwise miss happens when the particle sets of two neighbouring nodes are too *different*, where difference of particle sets can be characterised by the lack of similar particles amongst both sets. Assuming that a pairwise term ψ_{st} encourages smoothness, $\psi_{st}(\mathbf{u}_s, \mathbf{u}_t)$ will be high when $\|\mathbf{u}_s - \mathbf{u}_t\|$ is low (e.g. when $\mathbf{u}_s$ is spatially close to $\mathbf{u}_t$), and inversely will be low when $\|\mathbf{u}_s - \mathbf{u}_t\|$

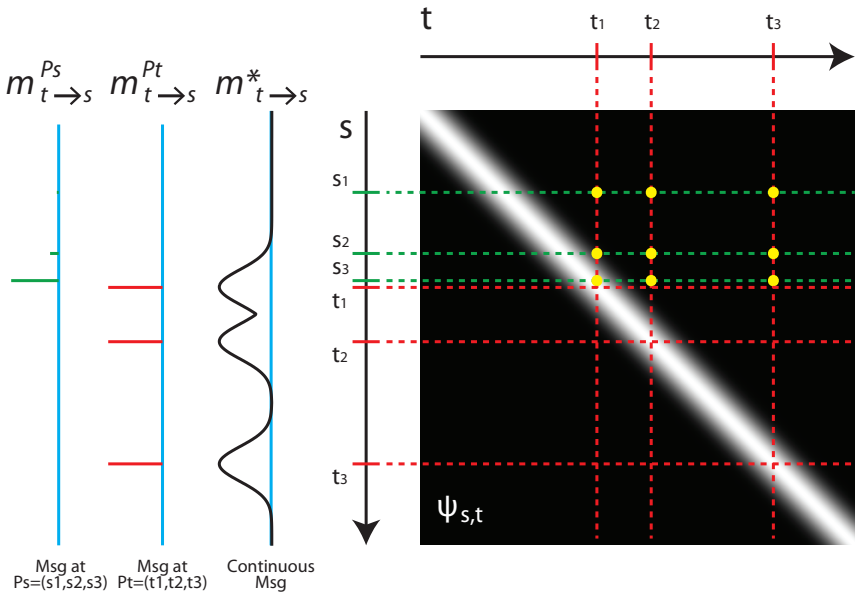


Figure 3.6: Message calculation and illustration of the "pairwise miss" phenomenon. Green bars represent the set of particles at s, $P_s = (s_1, s_2, s_3)$ and the red bars represent $P_t = (t_1, t_2, t_3)$. In PBP [Ihler et al., 2009, Kothapa et al., 2011], the continuous message $m^*_{t\to s}\mathbf{u}_s$ is evaluated only at particles in P_s, and minimized only over P_t, evaluated at the yellow dots. When P_s and P_t differ, much of the message may be uninformative (represented by the green particles $m^{Ps}_{t\to s}$). This is the "pairwise miss" phenomenon. If the pairwise potential favours smoothness, including particles from P_t increases the likelihood that high probability parts of the message are included, which is what happens in PMBP.

is high (i.e. when $\mathbf{u}_s$ is spatially far from $\mathbf{u}_t$). If the particles of both sets are too far away, no useful information will come out of the message, and the peaks of the real message will be missed, hence the term "pairwise miss". Figure 3.6 illustrates this phenomenon. In our work, we address this weakness in order to accelerate and improve the quality of the optimisation by making the link between PBP and the PatchMatch algorithm, and then combining the two to obtain an algorithm superior to both.

3.4 The PatchMatch Algorithm

The **PatchMatch** algorithm [Barnes et al., 2009] was initially introduced as a computationally efficient way to compute a *nearest neighbour field* (NNF) between two images. The NNF is then used for image editing operations such as denoising, inpainting or deblurring. In terms of energy minimisation, the NNF is the global minimiser of an energy comprising a unary term only ($\psi_{st} = 0$). The PatchMatch algorithm computes good minima while evaluating the unary term many fewer times than the number of possible states in the search space. With such a powerful optimiser, more complex unary terms can be defined, yielding another class of state-of-the-art correspondence finder, such as [Bleyer et al., 2011]. We describe in detail below how the algorithm works.

3.4.1 Methodology

PatchMatch is a randomised, iterative algorithm that uses the properties of natural images to accelerate the correspondence field estimation. We first describe the algorithm in the original terms in which it was published, and then in terms that will allow easy unification with standard description of continuous-domain BP, using particles.

The original formulation

The algorithm consists of three main components: an initialisation step, which is run only once, and then two alternating steps which are propagation and randomisation, that are run iteratively until convergence.

Initialisation The NNF is randomly initialised using a uniform probability distribution to sample from the search space. Alternatively, it can be initialised using prior knowledge about the NNF.

Propagation Natural images tend to be locally consistent, which means that good matches are often clustered into contiguous areas. Using this property, this step tries to improve the NNF by propagating the good matches between neighbouring patches. For example, given a patch p at (x, y), we consider its neighbour p_{left} located at $(x-1, y)$. If there is a good match (u, v) at p_{left}, we can assume that is it also likely that $(u+1, v)$ is a good match for p. In other words, if $\text{NNF}(x-1, y) = (u, v)$, then is likely that $\text{NNF}(x, y) = (u+1, v)$.

Randomisation This steps aims at improving the existing matches by performing a random search around each best match. To improve this match, we compute the similarity between the current patch and a sequence of random candidate patches located at an exponentially decreasing distance from the best match so far. This way the matching process for each patch is able to escape from local minima, as the first candidates are relatively far from the current best match. Moreover, the location of a match that is already good can be further refined, as we also consider candidates that are located at a small distance.

The propagation and randomisation steps are run on each pixel in scanline order starting from the top-left to the bottom-right of the image on odd-numbered iterations, and in the reverse order on even-numbered iterations. The algorithm runs until the halting criterion is met, which can be for example a threshold on the total error over the image. At first sight, it may appear surprising that such a simple algorithm can effectively minimise energies such as $\sum_s \psi_s^{\text{wpf}}$, but as the analysis in [Barnes et al., 2009] shows, the piecewise smoothness in typical image flow fields[1] effectively shares the optimisation burden among neighbouring pixels in the same smooth segment, without any need to identify those segments in advance.

PatchMatch with particles

In order to introduce a common formulation of PatchMatch and Belief Propagation, we consider the three main steps of PatchMatch in terms of particles.

Notation To simplify the descriptions below, we use the following notation. Define the application of a function f to a set S by $f(S) := \{f(s)|s \in S\}$. Define the function $\text{fargmin}_K(S, f)$ as the function that returns the K elements of S which minimise f:

$$\text{fargmin}_K(S, f) := S_K \subset S \quad \text{s.t. } |S_K| = \min(K, |S|) \text{ and } \max f(S_K) \leq \min f(S \backslash S_K) \tag{3.11}$$

With each node s, we associate a set of K *particles* $P_s \subset \mathbb{R}^d$, where each particle $p \in P_s$ is a candidate solution for the minimising correspondence parameters $\mathbf{u}_s^*$.

Initialisation The initialisation consists in sampling uniformly random particles to build the sets at each node.

One PatchMatch iteration then comprises a linear sweep through all nodes. The order in which nodes are visited is defined by a *schedule function* $\phi(s)$, so that s is visited before s' if $\phi(s) < \phi(s')$. We also define the *predecessor set* $\Phi s = \{s'|\phi(s') < \phi(s)\}$. If $iter$ is an iteration counter, we write $\phi_{iter}(\cdot)$ to select the appropriate schedule. At node s, two update steps are performed: *propagation* and *resampling*:

Propagation In this step, the particle set is updated to contain the best K particles from the union of the current set and the set C_s of already-visited neighbour candidates

$$C_s = \bigcup \Big\{ P_t \;\mid\; t \in N(s) \cap \Phi s \Big\}, \tag{3.12}$$

where "best" is defined as minimising the unary cost $\psi_s(\cdot)$:

$$P_s \leftarrow \text{fargmin}_K(P_s \cup C_s, \psi_s). \tag{3.13}$$

Resampling The local resampling step perturbs the particles locally according to a proposal distribution. In the original PatchMatch algorithm, the resampling is done using concentric balls within the search space, we can be modelled as a piecewise constant distribution with discontinuities at the boundary of

[1] Note that "flow field" is intentionally left imprecise here. The key is that the globally optimum NNF is *not* smooth, but the approximate NNF found by PatchMatch tends to be, due to the propagation step which takes advantage of the smoothness of the underlying real-world physical process to generates the image correspondences.

each ball. Alternatively, we can model the resampling distribution as a Gaussian $\mathcal{N}(0, \sigma)$. The second step of the PatchMatch iteration updates P_S with any improved estimates from the local resampling set, for m resampling steps:

$$R_s = \{p + \mathcal{N}(0, \sigma) \mid p \in P_s\} \tag{3.14}$$

$$P_s \leftarrow \text{fargmin}_K(P_s \cup R_s, \psi_s). \tag{3.15}$$

After several alternating sweeps, the best particle in each set typically represents a good optimum of the unary-only energy.

3.4.2 Limitations

A key deficiency of PatchMatch is that it lacks an explicit regularisation control on the output field. Indeed, recent developments of PatchMatch have noted that PatchMatch "has difficulty finding reliable correspondences in very large smooth regions" [HaCohen et al., 2011]. [He et al., 2011a] require a smooth field when applying PatchMatch to an alpha matting problem, and therefore regularise via a post-process step, by solving the matting Laplacian. [Boltz and Nielsen, 2010] achieve smoothness by dividing the images into super-pixels and running PatchMatch on these, meaning that a failure of super-pixelisation cannot be recovered from.

A related deficiency is the tendency of PatchMatch to require a form of "early stopping": the global optimum of the unary energy is not necessarily the best solution in terms of image error, as we show in figure 3.10h, and as can be seen in a figure of [Mansfield et al., 2011]. These difficulties are exacerbated by more powerful PatchMatch algorithms [Barnes et al., 2010, Korman and Avidan, 2011] which, although getting closer to the globally optimal NNF, lose the implicit smoothness that early stopping provides. An example of this behaviour is mentioned in [Barnes et al., 2010], where the attempt at denoising an image with PatchMatch shows that it tends to actually match the noise itself. We characterise this tradeoff by looking at *error* versus *energy*: the correlation between ground-truth errors (e.g. peak signal-to-noise ratio (PSNR) for denoising problems, end-point error (EPE) for 2D correspondence fields, or disparity error for stereo) and the values of the energy functions the algorithms implicitly or explicitly minimise. We see that PatchMatch optimises an energy that is not suitable to the problem - we can look at the example shown in figure 3.10h. While the unary energy, which PatchMatch is explicitly minimising, decreases, the PSNR increases after reaching a minimum early in the process. We also show the energy including a regularisation term, and use the solution that PatchMatch produces to evaluate it, and see that similarly to the PSNR, it starts increasing as well after reaching a minimum early on. In section 3.7.1 we carry out more experiments to analyse this behaviour and expose the limitations of PatchMatch.

3.5 Our Method: PatchMatch Belief Propagation

We define a new algorithm, called PatchMatch Belief Propagation (PMBP), which combines the best features of both existing approaches, and which includes the existing methods as special cases. We consider PBP our base, as the goal is to minimise a more realistic energy than PatchMatch, that is to say,

Let P_s be the set of particles at node s, and K the desired number of particles.
Let N be the number of iterations, and m the number of randomisation iterations.
Let $\mathcal{I}$ be the initialisation distribution: uniform or local potentials ψ.

PM	PBP	PMBP		Steps
•	•	•	1	**for all** nodes $s \in \{1..n\}$, **repeat** K times: // initialisation
•	•	•	2	Draw $p \sim \mathcal{I}$, add to P_s
•	•	•	3	**for** $i = 1$ to N: // main loop
•	•	•	4	**for all** nodes $s \in \{1..n\}$ **orderby** ϕ_i: // PatchMatch schedule
•	•	•	5	**for all** proposal sets R_s in:
•	-	•	6	$R_s = \bigcup\{P_t \mid t \in N(s) \cap \Phi_i(s)\}$ // resampling using neighbours
•	•	•	7	$R_s = P_s$ // local resampling
•	•	•	8	**do**
•	•	•	9	**for all** particles $p \in R_s$:
•	•	•	10	**repeat** m times // Possibly different m for each R_s
•	•	•	11	$p' = p + \mathcal{N}(0, \sigma)$
•	-	-	12	Compute $B_s(p') = \psi_s(p')$
-	•	•	13	Compute $B_s(p') = \psi_s(p') + \sum_{t \in N(s)} M_{t \to s}(p')$
•	-	•	14	$P_s = \text{fargmin}_K(P_s \cup \{p'\}, B_s)$ // Update best K in P_s.
-	•	-	15	**if** $B_s(p') < B_s(p) - \log(\text{rand})$: $p \leftarrow p'$ // MCMC sampling
-	•	-	16	Replace p with p' in P_s // Only after MCMC
•	•	•	17	**for all** nodes $s \in 1..n$: // read out the final solution
•	-	-	18	return $\text{fargmin}(P_s, B_s)$ where $B_s(p) = \psi_s(p)$
-	•	•	19	return $\text{fargmin}(P_s, B_s)$ where $B_s(p) = \psi_s(p) + \sum_{t \in N(s)} M_{t \to s}(p)$

Table 3.1: **Pseudo-code for different algorithms**. PM is PatchMatch; PBP is Max Product Particle BP; PMBP is PatchMatch BP. Note that whenever B_s is computed, for PBP and PMBP, we have to also recompute the minimizations in the messages $M_{t \to s}$.

an energy with pairwise terms that encourage smoothness in order to regularise the solution.

3.5.1 Components

Our algorithm has two main components: a resampling step using the particles of the neighbours, and a simple but efficient strategy to update the particle sets.

Neighbour resampling First, PM resamples P_s from the neighbours of node s, while PBP uses MCMC to resample the elements of P_s. As illustrated in figure 3.6, this may be viewed as sampling from the continuous incoming messages at s, with the property that important modes of the belief may be uncovered, even when P_s lacks particles at those modes. It should be clarified that the samples are evaluated using B_s, so this is a resampling of the particle set under the current belief, as proposed in PBP, but with a quite different source of particle proposals. Thus PMBP augments PBP with samples from the neighbours (or, as argued in Figure 3.6, samples from the incoming messages). This can also be viewed as a return to the sampling strategies of NBP [Sudderth et al., 2002], but with a much simpler message representation. One way to look at this contribution is simply to say we are running some form of NBP but with algorithm settings (number of particles, number of samples) that would never make sense for NBP, and that this in itself is a useful contribution. Note that taking directly particles from the neighbouring nodes only works because our pairwise term is a smoothing term, i.e. has the lowest value when both entries are the same. Hence for arbitrary pairwise terms this strategy has to be modified, and it can be generalised that the particle should be chosen as the one that minimises the current pairwise term.

Keep k-best strategy Second, PBP uses an MCMC framework where particles are replaced in P_s with probability given by the Metropolis acceptance ratio, while PatchMatch accepts only particles with higher belief than those already in P_s. We call this strategy "keep k-best" as we automatically select particles with the best associated beliefs. We have found that this non-Metropolis replacement strategy further accelerates convergence without degrading the quality of the solution, so it is included in PMBP. Our experiments carried out in section 3.7 support our claims.

3.5.2 Flexibility

Making these two modifications yields "PatchMatch BP", a powerful new optimisation algorithm for energies with pairwise terms. In the case of a zero pairwise term $\psi_{st} = 0$, PMBP exactly yields Generalised PatchMatch. More generally, a constant pairwise-term has the same effect. To illustrate this mechanism, we consider a the pairwise-term that is equal to a constant value $\psi_{st} = c$. Equation 3.10 becomes:

$$M_{t\to s}(\mathbf{u}_s) = \min_{\mathbf{u}_t \in P_t} c + B_t(\mathbf{u}_t) - M_{s\to t}(\mathbf{u}_t) = c_t \tag{3.16}$$

where c_t is a constant depending on the particle set P_t. That means that the message incoming at node s is constant, and therefore carrying no information that would lead to favouring any of the particles. Therefore, the disbelief at node s becomes:

$$B_s(\mathbf{u}_s) = \psi_s(\mathbf{u}_s) \; + \sum_{t \in N(s)} c_t = \psi_s(\mathbf{u}_s) \; + c_t. \tag{3.17}$$

As a result, the disbelief is directly related to the data term, and in fact *is* the data term shifted by a constant that does not depend on the value of u_s. To be able to control the level of regularisation of the solution, we use the pairwise weight λ mentioned in equation 3.1. In practice, this consists in weighting the pairwise term ψ_{st} seen in equation 3.10 with λ. Setting $\lambda = 0$ yields a constant (and null) pairwise term which produces the behaviour described previously: PatchMatch.

Conversely, running PMBP with a nonzero pairwise term is a strict generalisation of GPM, allowing the incorporation of an explicit regularisation control which directly addresses the deficiencies of PatchMatch while retaining its speed. Moreover, the strength of the pairwise term is modulated with λ, which allows for a fine control over the algorithm's behaviour as it makes the relative influence of the data term more or less important. The convergence rate of the PBP algorithm is $O(1/\sqrt{n})$ where n is the number of samples. Since the PMBP algorithm is an augmentation of PBP, we expect its convergence rate to be at least faster than that of PBP.

3.5.3 A Modular Representation

We describe the steps of PatchMatch, PBP and PMBP in table 3.1. These three algorithms can now be represented within the same unified framework. In this table, we can find components of both PatchMatch and Belief Propagation in combination. We will point out here the similarities and differences.

Similarities PM, PBP and PMBP have a large number of common steps. Out of 19 steps described in the algorithm, 11 are common to the 3 algorithms. They all use a random initialisation of the particle sets (lines 1 and 2), and are both iterative algorithms (line 3) using a scheduling function (line 4) to process the nodes in order. The random resampling is also a common step (lines 7 and then 11 and 12), as new particles are drawn in the same way.

Differences A few steps differentiate the 3 algorithms and make them unique. First, the resampling mechanism using the neighbours is only used by PM and PMBP (line 6). Second, while PM uses only the data term (line 12), both PBP and PMBP performs a full evaluation of the disbelief using the incoming messages (line 13) although this could also be unified using the λ pairwise weighting factor and attributing the value $\lambda = 0$ to PM. The same point can be made for the operation returning the solution (lines 18 and 19). Finally, the MCMC step is present in PBP (lines 15 and 16) but is not used in either PM and PMBP (line 14).

Note that as an extra step, we can also use any external information to get reasonable candidate particles, such as matching nodes between image pairs in the stereo matching case, as we will describe more in detail in Chapter 4.

3.6 Implementation details

While the algorithm is described in table 3.1, there are some implementation issues that are worth describing, as well as their solutions. These points are also described in [Besse et al., 2013b].

3.6.1 Caching

First, PMBP and in general all loopy BP algorithms are defined in a recursive manner. Equation 3.9 illustrates this, as the computation of a message at one node depends on the messages at other nodes.

Therefore, the usual method is to proceed in an iterative fashion, where the new messages are computed from the messages calculated in a previous iteration. This implies that the messages need to be stored for later use. To illustrate this, let us explicitly label messages and particle sets with the iteration number at which they are computed. We consider the message from node t to s at iteration k, $M_{t\to s}^k$. We call P_s^k the particle set at node s at iteration k. We also call B_s^k the disbelief at node s at iteration k, and it is computed as follows:

$$B_s^k(\mathbf{u}_s) = \psi_s(\mathbf{u}_s) \; + \sum_{t \in N(s)} M_{t\to s}^{k-1}(\mathbf{u}_s), \tag{3.18}$$

which is equivalent to equation 3.8, except that we explicitly indicate which messages need to be used.

Having introduced these notations, we can now see that there is a conceptual problem with respect to storing previously computed messages, as messages are a functions of the receiver state. In a nutshell, this is due to the fact that the particle set at each node changes continuously due to the resampling procedure, which invalidates previously stored messages as they can only be evaluated on the set of particles that was present at the time they were stored. This can be easily seen in equation 3.18: the belief needs to evaluate several messages at particle position $\mathbf{u}_s$, which might not have been part of the previous particle set P_{k-1}; in other words the values of the message at $\mathbf{u}_s$ might simply not exist.

To resolve this issue, let us consider how the messages are computed. We can rewrite equation 3.10 as follows:

$$M_{t\to s}(\mathbf{u}_s) = \min_{\mathbf{u}_t \in P_t} \; \psi_{st}(\mathbf{u}_s, \mathbf{u}_t) + m_{t\to s}(\mathbf{u}_t). \tag{3.19}$$

with

$$m_{t\to s}(\mathbf{u}_t) = B_t(\mathbf{u}_t) - M_{s\to t}(\mathbf{u}_t). \tag{3.20}$$

We call the object $m_{t\to s}$ the "pre-message" or "message foundation" from node t to s. This object has an interesting property: it is a function of the sender node and not of the receiver. It contains information about t and is expressed in terms of particles of t. It is converted through equation 3.19 into $M_{t\to s}(\mathbf{u}_s)$, i.e., information about s and is expressed in terms of particles of s. As the pairwise term ψ_{st} is known, our caching strategy can be reduced to caching the message foundations only. As they are not tied to the particle states of the receiver, and depend *only* on the state of the sender, it is suitable for our framework. In practice, after having updated the particle set of node t at a given iteration, we compute all the message foundations outgoing from t and store them for later use, before continuing to the next node according to the schedule.

3.6.2 Existing particles update

Another implementation detail that is important to consider is the update of the disbelief at the previously existing particles of a set. When following the schedule and reaching a node s, the particles of the current set P_s are associated with existing disbelief values that were computed at the previous iterations. These values were computed using the messages that existed at the previous iteration, which might now have changed. Therefore, the first step that is run when reaching a node is to *recompute* the disbelief values for the existing set of particles, so that they reflect the new changes of the incoming messages.

3.6.3 Normalisation

As mentioned in [Nowozin and Lampert, 2011], using a max-product formulation creates a numerical instability as large numbers are formed from accumulation of the messages. This happens very quickly, as the growth is exponential. To keep the algorithm stable, a normalisation if performed on the message foundations whenever they are recomputed, which consists in shifting the values so that they all sum to zero. That is, assuming that $\bar{m}_{t\to s}$ is the non-normalised message foundation from t to s, we have:

$$\delta = \frac{1}{|P_t|} \sum_{u_t \in P_t} \bar{m}_{t\to s}(u_t) \tag{3.21}$$

$$m_{t\to s}(u_t) = \bar{m}_{t\to s}(u_t) - \delta \tag{3.22}$$

3.6.4 Early termination

The original PatchMatch algorithm included an early termination step, which consisted in regularly checking the current value of the error within the error calculation loop, and terminating early if the current value was already worse than that of the best candidate so far. This was straightforward as the fitness of a candidate was only dependent on this unary error calculation coming from the data term. In our case, we use both unary and pairwise terms, and the suitability of a particles depends on both associated energies, as it is the full disbelief that is used to decide whether a particle should be included in the set of not. However, we can still perform an early termination check.

To do so in an efficient way, we shall remember that the evaluation of the data term is often much more expensive than that of the messages. Indeed, in a patch matching situation, comparing 2 patches of size n composed of RGB pixels will involve $3n^2$ pixel comparisons, when using colour information only. On the other hand, the evaluation of the pairwise term often only depends on the state itself, and is closely related to its dimensionality. For example, in the 2D-PCE (see figure 3.2), while the unary term require the computation of $3n^2$ pixel differences, and a global summation, the pairwise term only involve the comparison of two values, which means that the message summation is performed over N values, with N being the number of messages summed, with each message involving a minimisation over n values where n is the number of particles used. For relatively low number of particles, the message computation and summation is much faster executed, and the minimisation can be accelerated using suitable data structure, such as trees. Having this in mind, we can efficiently perform an early termination on the disbelief evaluation. Let us call B_s^* the disbelief of the "worst" particle in P_s. If a new candidate particle has a disbelief that is superior than B_s^*, we known that it is not going to be accepted into the set. Then, we can first compute the message summation, and then run the early termination on the unary term by using as a threshold of τ where:

$$\tau = B_s^* - \sum_{t \in N(s)} M_{t\to s}(\mathbf{u}_s). \tag{3.23}$$

as this leads to:

$$\begin{gathered}\psi_s(\mathbf{u}_s) > \tau \\ \Leftrightarrow \\ \psi_s(\mathbf{u}_s) > B_s^* - \sum_{t \in N(s)} M_{t \to s}(\mathbf{u}_s) \\ \Leftrightarrow \\ B_s(u_s) > B_s^*\end{gathered} \tag{3.24}$$

which is a satisfactory condition for the early termination of the disbelief evaluation.

3.7 Experiments

In this section we present several experiments aiming at showing the mechanisms, performances, benefits and limitations of PatchMatch, Belief Propagation, Particle Belief Propagation and our algorithm PMBP. We first run a series of experiment to analyse more in detail the PatchMatch algorithm, and show its weaknesses and compare it to PMBP. We then proceed to a similar analysis of the Belief Propagation, and in turn Particle Belief Propagation showing each algorithm's behaviour and limitations compared to PMBP. Finally, we analyse PMBP itself via a series of experiments, and introduce its limitations.

3.7.1 PatchMatch

As discussed previously, there are three main components of the PM algorithm: the random initialisation, the propagation and the randomisation. We conduct a series of experiments to assess the importance of each of these steps, which show two things. Indeed, the propagation is the most important step yielding the most significant drop in energy, but needs the randomisation to be able to evolve past local minima. We will later on, in the PMBP experiments, show that the propagation is once again the most crucial part of the algorithm.

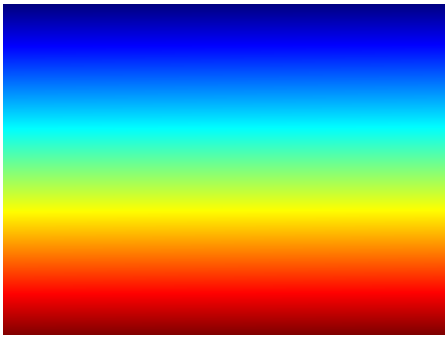
Propagation and randomisation

PM relies on the likelihood that some initial guesses of the states will be correct and propagate to neighbouring pixels. As explained below, this is the reason that one can benefit from constraining the search space as much as possible in order to have more chances for these random good matches to occur. To actually see the importance of these guesses on the match field, we tag each pixel with an integer from 1 to n where n is the number of pixels in the image. At the initialisation step, each random state that we assign to each pixel carries the id of its pixel. We call those states "seed states". Assigning IDs to seed states allows us to "track" them through the process. When a state is overwritten due to the propagation of a better state coming from a neighbour, we copy over the ID of this better state to be able to track where seed states are propagated. Randomisation does not affect the ID of a seed sate. We represent the ids as a colormap, and show this representation directly after initialisation and after one iteration. We also show the evolution of the total number of different seed states in the image. This can be seen in figure 3.7.

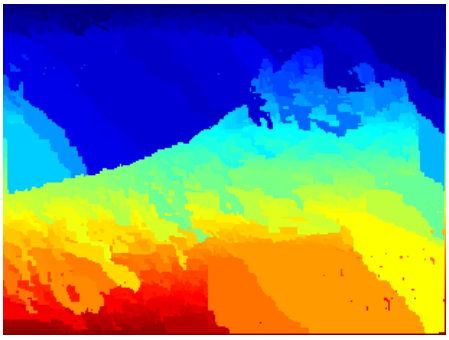
First, we note that at initialisation there are 70610 different seed states in the image (one per pixel), after one iteration this number drops to 2351, then to 1106 after two iterations to slowly reach 943 after

(a) Source image

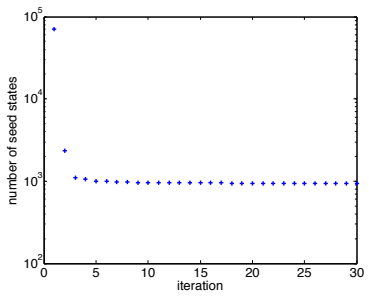
(b) Target image



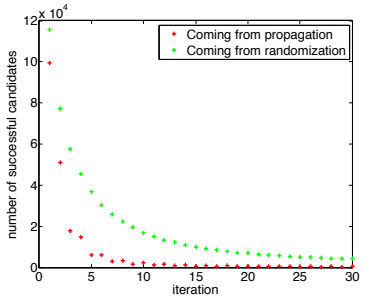
(c) Initial states ids



(d) States ids after 1 iteration

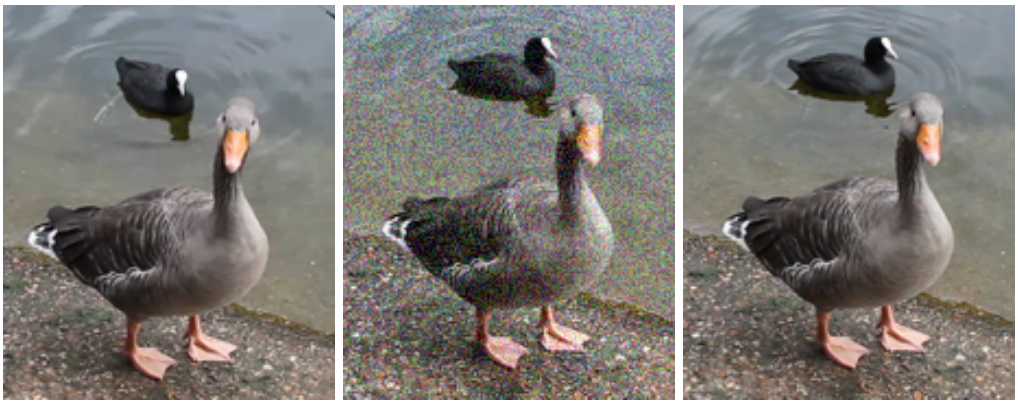


(e) Seed state population



(f) Number of successful candidate states coming from propagation and randomisation

Figure 3.7: Matching of the two images shown in (a) and (b). (c) shows that each pixel is assigned an ID. After one iteration, we track the IDs and show them in (d). This effectively track the seed states (or initial states) through the matching process. (e) shows the evolution of the number of seed states present in the image, and (f) shows the number of successful state assignment due to propagation and randomisation.



(a) Source image (b) Target image (c) Ground truth

Figure 3.8: Denoising application. The goal is to denoise the target image, by using patches from the source image.

30 iterations which represents only 1.3% of the original states. The plot of this number is shown in semi-log scale on the Y axis because of the large drop in the first iteration. This illustrates the importance of the propagation, and the fast convergence rate that PM usually exhibits. In [Barnes et al., 2009], the authors typically use a small number of iterations, which can be related with the aggressive propagation happening during these iterations. We can conclude that the propagation is crucial during the first iterations of the algorithm, and its usefulness decreases later on, where the only chances of improving the solution lies with the random search. This can be seen in figure 3.7f, where we show the number of times that a state candidate has successfully improved the current particle set, coming from the propagation and the randomization distinctively. Here the propagation curve decreases more rapidly than the randomisation curve, which illustrates the point that was just discussed. We can also see in figure 3.7e that the size of the areas of same seed states can spread to a significant portion of the image, creating diagonal streaks. This pattern is due to the scheduling function, which in our case is in scanline and reverse scanline order. Iterating on pixels by following a schedule which processes adjacent pixels in order favours the efficiency of the propagation.

3.7.2 Limitations of PatchMatch and comparison with PMBP

While the performance of PM and its benefits compared to other standard algorithms has been demonstrated in [Barnes et al., 2009], we conduct another series of experiments to show where the algorithm fails and could be further improved. In particular, we show that for applications requiring smoothness, such as denoising, when using a complete energy with both unary and pairwise terms, the error produced by PM reaches its minimum early during the iterative process, and then increases as without pairwise terms it is essentially matching unwanted noise.

Denoising application 1

For this experiment we create a simple denoising application. We use the 2D-PCE between two different images of a scene, one noisy (Gaussian noise was artificially added) and one clean. The images can be

seen in figure 3.8. The goal is to remove the noise of the first image by using similar clean patches from the second image. Note that we do not know the ground truth displacement between the two images, so our assessment of the quality will be done by calculating the Peak Signal Noise Ratio (PSNR) between the noisy image and the clean version only. The PSNR between a $m \times n$ noise-free image I_1 and a noisy approximation I_2 is defined as:

$$\text{PSNR}(I_1, I_2) = 10 \log_{10} \frac{\max {I_1}^2}{\text{MSE}(I_1, I_2)} \tag{3.25}$$

with

$$\text{MSE}(I_1, I_2) = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I_1(i,j) - I_2(i,j)]^2 \tag{3.26}$$

and it compute the quality of reconstruction of an image, which is often used as measure of the quality of compression algorithms.

To reconstruct the image, we simply cumulate the best matching patches into a new image, and compute the average values of all intensities, as performed in [Barnes et al., 2009]. We use 3x3 patches and run both PM and PMBP with regularisation enabled for 100 iterations with 5 particles. The results are visualised in figure 3.9. We can see that the result of PM is more noisy, which is due to the fact that without regularisation, the algorithm is essentially matching the noise. PMBP manages to get a higher PSNR with a smoother solution. The motion field of both algorithms can also be seen in the same figure, where the difference in smoothness is apparent. The PSNR over time of the two algorithms can be seen in figure 3.9. An interesting result is that PM obtains its best result early in the process, after which the PSNR starts decreasing, as discussed above. This is due to the propagation step that early on in the process results in many smooths regions across the image, which are slowly broken as the algorithm starts finding better optima somewhere else in the image due to the randomisation process.

Denoising application 2

We run a similar type of application where the goal is to recover a known displacement between two images. To do so, we select a clean source image, to which we apply a synthetic displacement field $u_s^{gt} = [\sin(x_s), 0]^\intercal$, and 10% Gaussian noise. To assess the results, we calculate the End Point Error (EPE) between the recovered flow and the ground truth as follows:

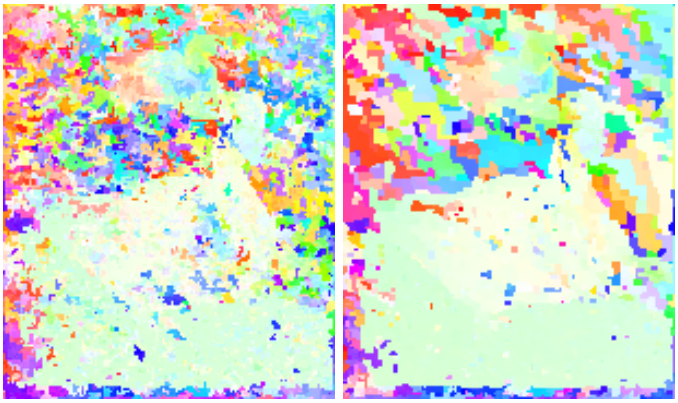
$$\text{EPE} = \sum_s \|\mathbf{u}_s - \mathbf{u}_s^{\text{gt}}\|^2. \tag{3.27}$$

The dataset and results can be seen in figure 3.10. Our method performs considerably better for both error measures in this task. The difference between (e) and (g) is especially noticeable in the smooth, green background where PatchMatch suffers from the ambiguous data term. (h,i) Plots error and energy for PatchMatch and our method. It is noticeable that the full energy with pairwise terms is a much better fit for the task, since in (i) both error measures are well correlated with the regularised energy, in contrast to (h), where the error curves increase as the PatchMatch iterations decrease the unary-only energy.

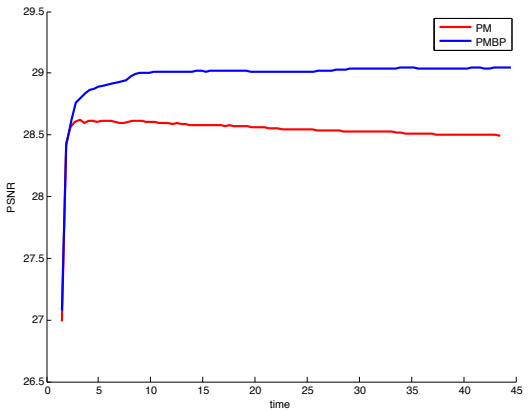
3.7.3 Belief Propagation

As mentioned previously, Belief Propagation is a rather slow algorithm when used on real-world applications that involve images due to the size of the search space. We have seen that even on a discrete

(a) Ground Truth (b) PM result (PSNR=28.49) (c) PMBP result (PSNR=29.04)



(d) PM motion field (e) PMBP motion field

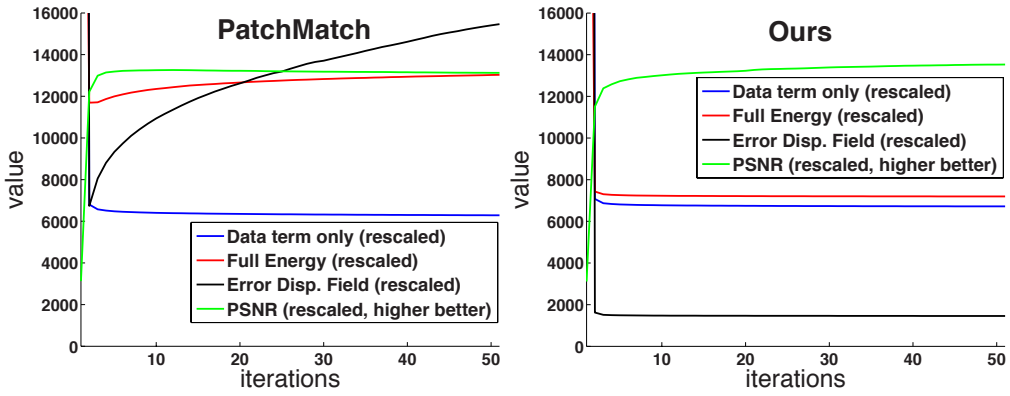


(f) Denoising PSNR over time for PM and PMBP.

Figure 3.9: Denoising results. We see that PatchMatch tends to match the noise. This behaviour can be justified by the lack of regularisation term. On the other hand, our result achieves higher PNSR, due to the included smoothness which is crucial to the denoising process.

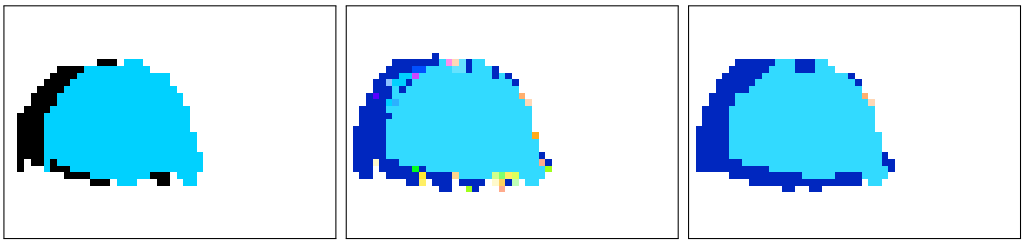
(a) Source (b) Displacem. field (c) Warped target

(d) EPE $= 4.63 \times 10^6$ (e) PSNR $= 37.39$ (f) EPE $= 0.44 \times 10^6$ (g) PSNR $= 38.36$



(h) PM: min of data term only (i) PMBP: min of full energy

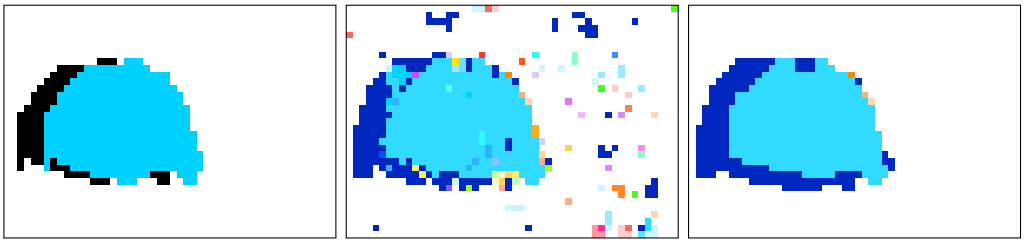
Figure 3.10: Example: denoising with a reference image. Note, red rectangle is a zoom of the top left corner. (e) and (g) show the reconstructed target image using PatchMatch and our method respectively.



(a) Ground truth

(b) Unary solution - no noise. Energy: 5988. Error: 0.18

(c) BP solution - no noise. Energy: 5528. Error: 0.14



(d) Ground truth

(e) Unary solution - added noise. Energy: 21324. Error: 0.61

(f) BP solution - added noise. Energy: 19540. Error: 0.25

Figure 3.12: Results on the toy example (a,b,c) and results of the toy example with added noise (d,e,f).

state space, the algorithm can quickly become computationally intractable. We designed an experiment to show the benefits of using BP in an application requiring regularisation compared to PatchMatch. In addition to this, we illustrate the difficulties caused by having a discretisation of space.

Toy example on a discretised space

(a) Source image

(b) Target image

(c) Ground truth

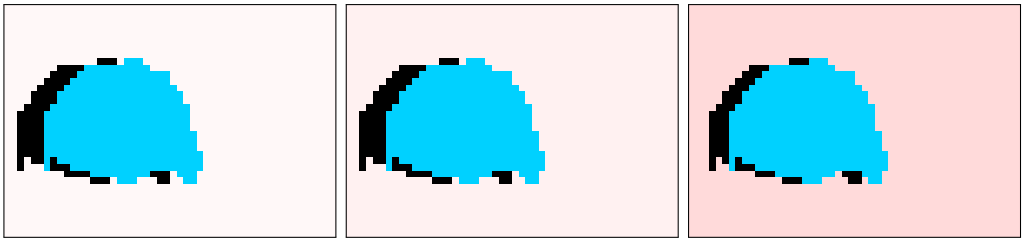
Figure 3.11: Toy example with artificial displacement and ground truth. Black areas on the ground truth image indicate areas where no ground truth is available.

We use a toy example where an area has been artificially displaced by 3 pixels in the x-direction. The target image, source image and ground truth is shown in figure 3.11. Note that the black areas in the ground truth indicate areas that there is no solution. To be able to run 2D-PCE with discrete BP, we discretise the search space into a grid of one pixel interval covering the integer displacement between -5 and 5 in both axes (11 values) for a total of 121 different states in 2 dimensions.

We note that this discretisation contains the true displacements values over the image, which are [-3,0] for the dog's head, and [0,0] for the rest of the image. We run BP on this image pair and visualise

(a) Target with 0.1 pixel shift (b) Target with 0.2 pixel shift (c) Target with 0.5 pixel shift

(d) Target with 0.1 pixel shift and noise (e) Target with 0.2 pixel shift and noise (f) Target with 0.5 pixel shift and noise



(g) GT with 0.1 pixel shift (h) GT with 0.2 pixel shift (i) GT with 0.5 pixel shift

Figure 3.13: A sub pixel shift is added to the target image on one set, and Gaussian noise is added in addition to the shift on a second set of images. While this might be difficult to see on the image itself, the pink shade indicating the ground truth displacement is visible on the ground truth images.

the result in figure 3.12[a,b,c]. We note that the regularisation term is beneficial as it helps retrieve a good solution at the patches forming the edges of the moving area, which do not exist in the source image. Next, we added Gaussian noise on the target image to better illustrate the benefits of the regularisation term. BP successfully finds a relatively noise-free solution compared to the solution given the lowest unary energy as seen on figure 3.12[d,e,f].

3.7.4 Limitations of Belief Propagation and comparison with PMBP

Conditions where discrete Belief Propagation is not suitable

In order to see the limitations of discrete BP, we add a sub-pixel shift to the whole image taken from the image with the dog example such that the discretised search space does not contain the true value of the displacement. We use a sub-pixel offset of -0.1, -0.2 and -0.5 and run BP on the images which can be seen in figure 3.13. We also increase the ambiguity of the problem by introducing noise in addition to the shift, which we show on the same figure as they the two sets of pictures share the same ground truth displacement. To establish a comparison between the algorithm we also run PM on these images and finally PMBP with the same smoothness settings as BP. The results are visualised in figure 3.14 for

the shift only and figure 3.15 for the shift with Gaussian noise. The main drawback of discretised BP is clear while looking at these results, as we see that the quality of the solution decreases as the ground truth solution drifts away from the reachable states of the discretised space. While the BP manages to find a relatively good solution compared to a pure brute-force search using the unary term only on the grid, for shift values of 0.1 and 0.2 in both experiments (shift and shift with noise), it fails when the shift value reaches 0.5. The results of PM show that having particles with variable positions is beneficial and it reaches a lower error than the BP results. However, the output images are visually not smooth due to the presence of noise. Finally, PMBP which combines both particle resampling and regularisation outperforms both methods and produce a smoother output than PM.

3.7.5 Limitations of Particle Belief Propagation and comparison with PMBP

While BP fails because of the discretisation of space, PBP uses particle resampling to address this problem. However, on more computationally demanding applications, such as those involving images, we show that the performance of PBP is lacking.

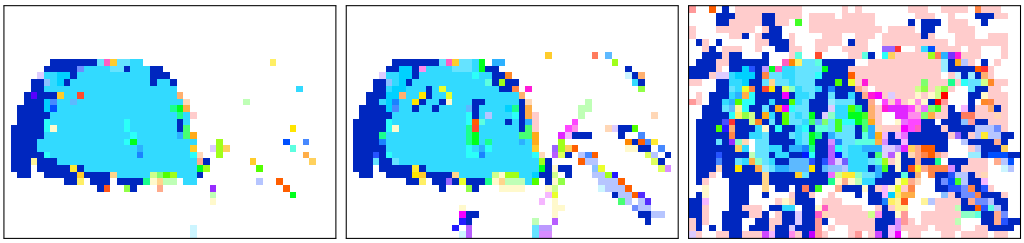
Denoising example

In order to examine the performance of PBP and PMBP on a large number of iterations, we design a similar experiment to that of section 3.7.2, but with smaller images. We run three algorithms with different variations:

- PM
- PBP with uniform initialisation
- PBP initialised with the result of PM, which we call "local potential initialisation"
- PMBP with uniform initialisation
- PMBP initialised with the result of PM
- PMBP with integrated MCMC resampling mechanisms.

All the algorithms were run with 10 particles.

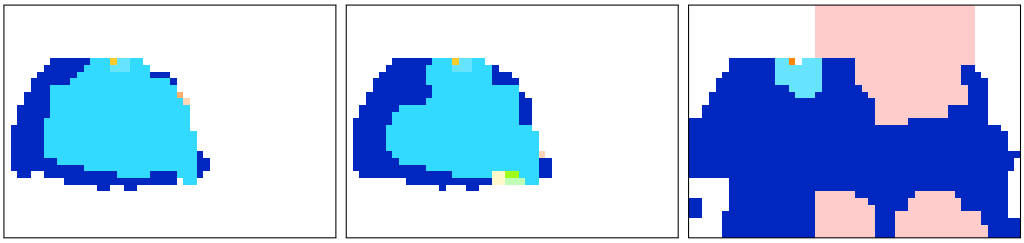
The results can be seen in figure 3.16. There are several important observations that can be made based on the results. First, the energy that is displayed on this curve is the full energy comprising unary and pairwise terms. We can see that once again PM does not optimise this energy and therefore it increases after a short period of time. Next, there is a clear performance difference between PBP and PMBP. Indeed, PMBP reaches a level of energy close to its final level as soon as $t = 1$, while PBP starts converging after four order of magnitude longer. Furthermore, the energy that PMBP converges to is much lower than that of PBP. We find that the initialisation only affects the quality of the result at earlier iterations. In other words, one can benefit from initialising with the local potential when having a limited time-budget to reach lower level of energies earlier than when using uniform initialisation. Finally, we can make a similar observation for the MCMC mechanisms. When using the *keep k-best* scheme rather than MCMC, PMBP reaches the final energy an order of magnitude faster. This also shows that the



(a) Shift 0.1 - unary solution. Energy: 13807. Error: 0.40.

(b) Shift 0.2 - unary solution. Energy: 29703. Error: 1.90.

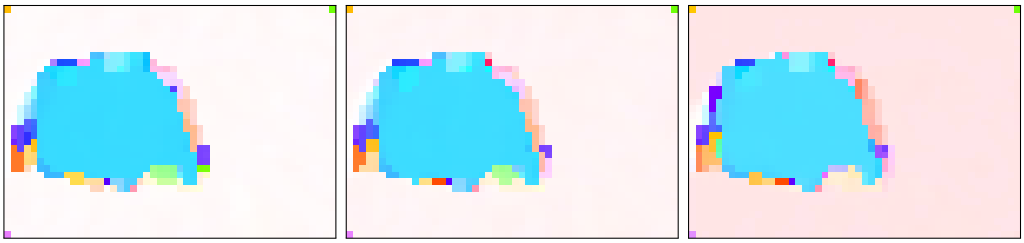
(c) Shift 0.5 - unary solution. Energy: 34790. Error: 3.46.



(d) Shift 0.1 - BP solution. Energy: 12775. Error: 0.28.

(e) Shift 0.2 - BP solution. Energy: 26113. Error: 0.66.

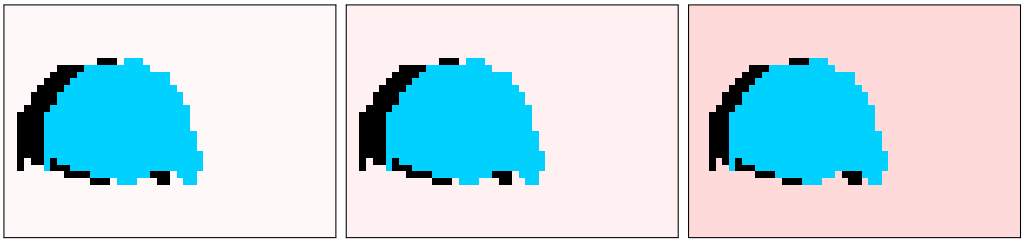
(f) Shift 0.5 - BP solution. Energy: 30862. Error: 5.38.



(g) Shift 0.1 - PMBP solution. Energy: 8756. Error: 0.06.

(h) Shift 0.2 - PMBP solution. Energy: 9972. Error: 0.07.

(i) Shift 0.5 - PMBP solution. Energy: 12320. Error: 0.10.

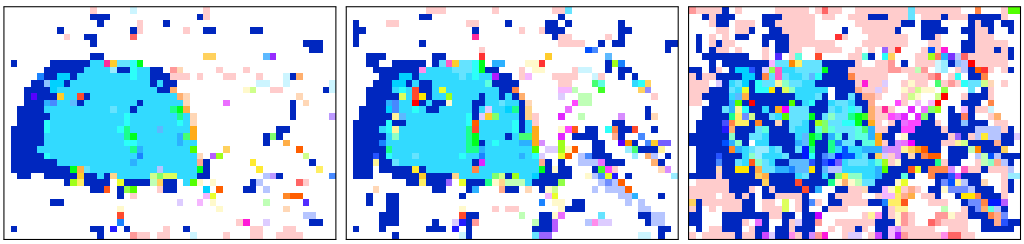


(j) GT with 0.1 pixel shift

(k) GT with 0.2 pixel shift

(l) GT with 0.5 pixel shift

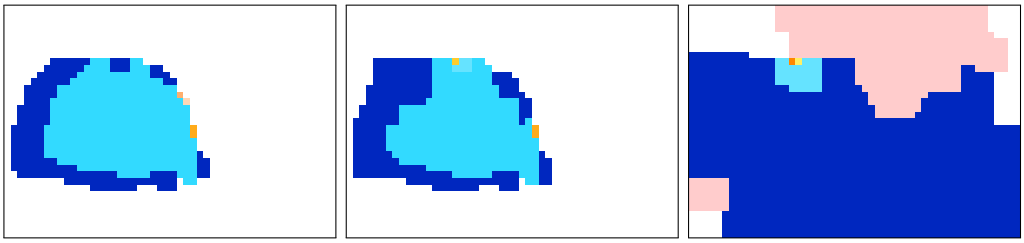
Figure 3.14: Results on the different shifted images for different algorithms.



(a) Noise + Shift 0.1 - unary solution. Energy: 24044. Error: 1.09.

(b) Noise + Shift 0.2 - unary solution. Energy: 29703. Error: 1.90.

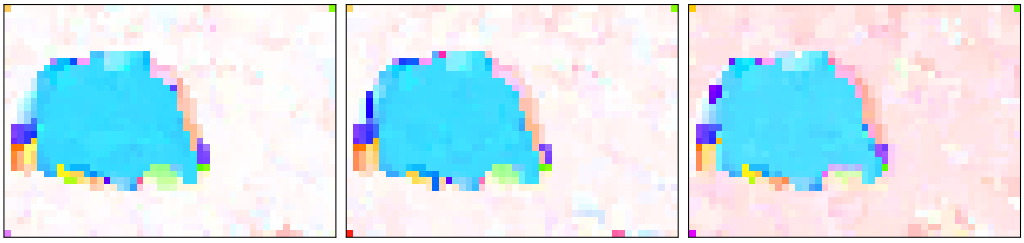
(c) Noise + Shift 0.5 - unary solution. Energy: 34790. Error: 3.46.



(d) Noise + Shift 0.1 - BP solution. Energy: 21646. Error: 0.37.

(e) Noise + Shift 0.2 - BP solution. Energy: 26113. Error: 0.66.

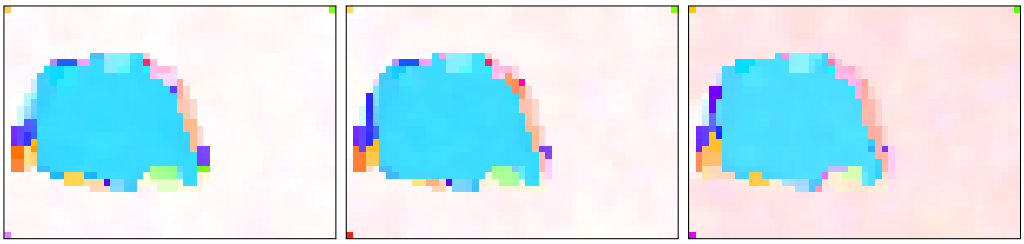
(f) Noise + Shift 0.5 - BP solution. Energy: 30862. Error: 5.38.



(g) Noise + Shift 0.1 - PM solution. Energy: 22605. Error: 0.21.

(h) Noise + Shift 0.2 - PM solution. Energy: 24232. Error: 0.32.

(i) Noise + Shift 0.5 - PM solution. Energy: 27231. Error: 0.79.



(j) Noise + Shift 0.1 - PMBP solution. Energy: 21447. Error: 0.10.

(k) Noise + Shift 0.2 - PMBP solution. Energy: 22050. Error: 0.13.

(l) Noise + Shift 0.5 - PMBP solution. Energy: 21959. Error: 0.41.

Figure 3.15: Result on the different shifted and noisy images for different algorithms.

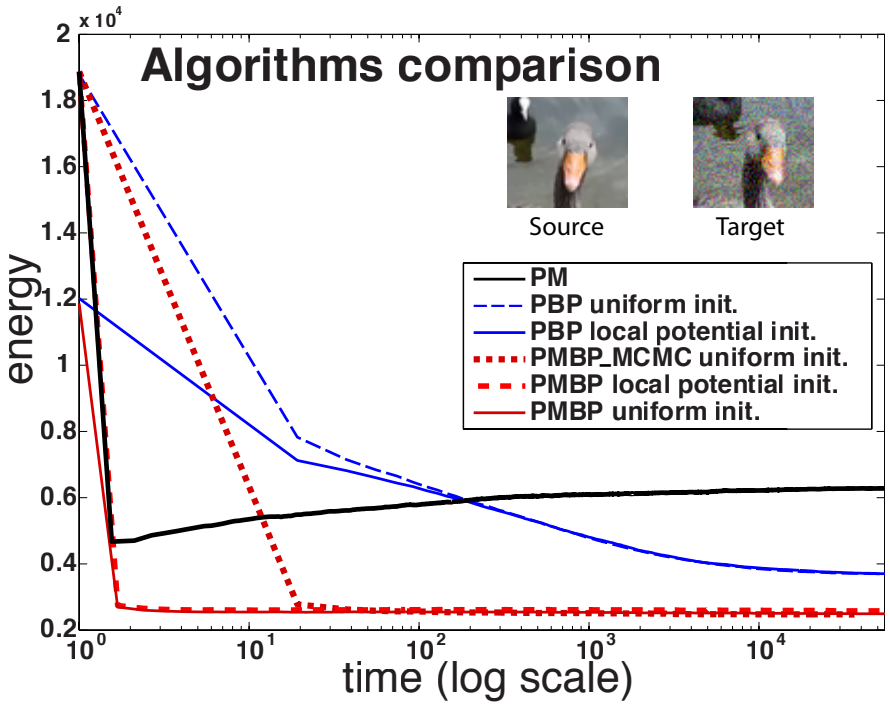


Figure 3.16: Comparison of the energies produced by the different algorithms on a denoising experiment. Notice that PBP cannot reach the energy of PMBP even if allowed four orders of magnitude longer, supporting our claim that previous BP implementations were intractable.

propagation is the most crucial step, as the difference between "PMBP MCMC" and "PBP" is *only* the propagation step.

We compare the evolution of the energy output by PMBP and PBP with the multi-label implementation of [Boykov et al., 2001] on a another small image denoising application. As GraphCut is a discrete method, we discretise the search space uniformly with d labels per pixel unit within the offset search window $[-5, 5] \times [-5, 5]$. We compare different levels of discretisation with our algorithm as well as PBP. Note that PMBP and PBP were run with 10 particles. Results can be seen in figure 3.17. Note that the plot is in semi-log space. We can see that PMBP converges faster and to a lower energy than all the other methods. The energy output by PBP is the second lowest after 1000s, and continue to slightly decrease as the algorithm keeps refining the particle positions, while GraphCut converges after a few iterations. The performance of the GraphCut depends on the level of discretisation. For a finer label grid, the final energy is lower, but the computational time is higher, as expected. In our experiment, the maximum discretisation level is set to $d = 5$ as the computation time becomes prohibitive for higher levels. Note that the GraphCut takes more time to initialise due to the evaluation of the unary and pairwise costs for each label. One iteration of PMBP takes about 0.1s, while one iteration of the GraphCut algorithm takes 0.14s, 1.08s, 4.54s, 13.20s and 32.38s for a discretisation factor of $d = 1$, 2, 3, 4 and 5 respectively.

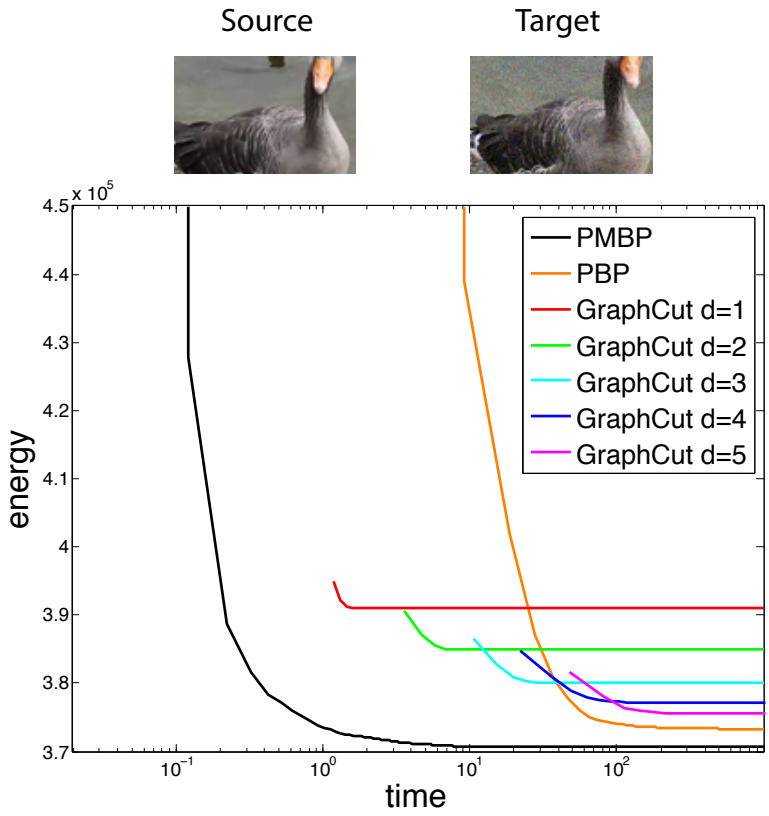
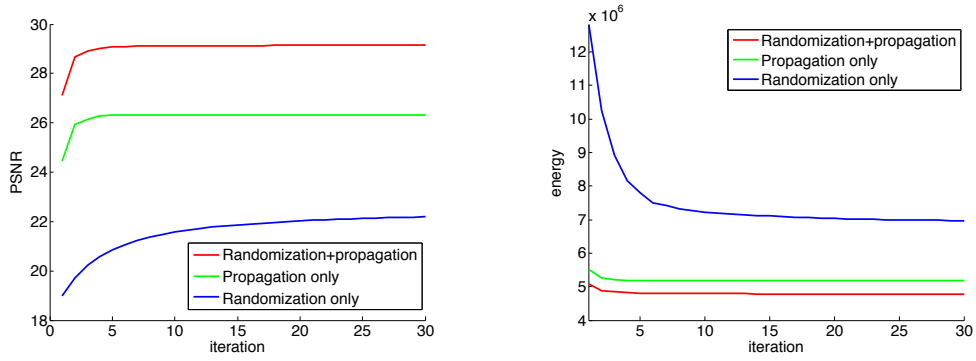


Figure 3.17: Comparison of the energy output by PMBP and PBP with a multi label GraphCut algorithm for different levels of discretisation of the label space. The coefficient d indicates the number of subpixel labels per pixel in the displacement search space.



(a) Evolution of the PSNR

(b) Evolution of the energy

Figure 3.18: Evolution of the PSNR and the energy over 30 iterations with different steps of the algorithm disabled.

3.7.6 PMBP

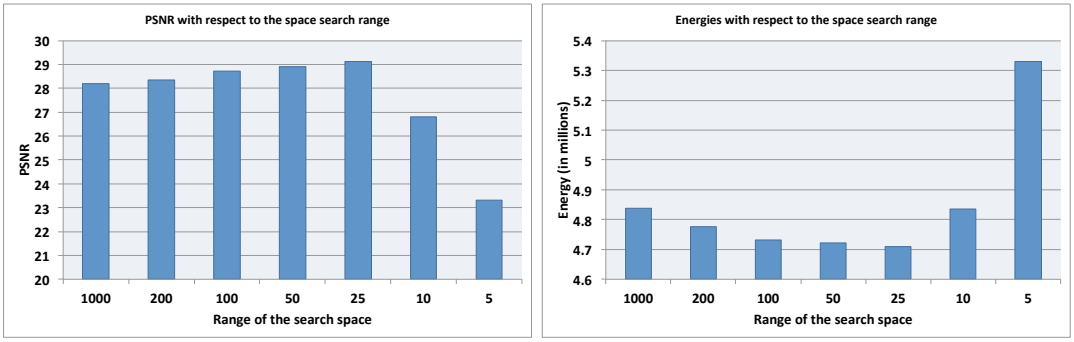
On a larger scale state-space, we have shown that PMBP is orders of magnitude faster than PBP, and is more accurate than PM in terms of the solution, as it optimises a more suitable energy. To illustrate the fact that the propagation is a crucial step, we run a series of experiments where we switch on and off several steps of the algorithm and compare the quality and performance of the results. We also show the influence of the size of the search space. This affects the manner in which the random initialisation is carried out, which is crucial to the performance of the algorithm, especially as dimensionality increases, and one can benefit from decreasing the size of the search space by using prior knowledge of the problem. This will be an important consideration in the following chapters of this thesis where we use high dimensional search spaces. and compare the performance with different number of particles. We examine the influence of the number of particles on the solution, with a particular focus on the textureless areas.

Propagation and randomisation

This experiment aims at showing the influence of the propagation and random resampling (which we call randomisation) of PMBP. We run the denoising application of section 3.7.2 over 30 iterations, and record the energies and PNSR at each iteration of the algorithm for 3 configurations: propagation only, randomisation only, and full PMBP comprising both steps. Results can be seen in figure 3.18. As expected the algorithm performs best when both steps are included. However it is clear that using the propagation only produce a result closer to that of the full algorithm compared to using the randomisation only.

Importance of the size of the search space

This experiment aims at showing the influence of the range of the search space on the quality of the results. To do so, we use the denoising example of section 3.7.2 (figure 3.8), use different ranges and plot the associated PSNR and energies after 10 iterations. Here the space is a disk, and the range that we display is the radius of this disk in pixel. Note that states that correspond to patches that are outside of the



(a) PSNR wrt search space range

(b) Energies wrt search space range

Figure 3.19: Evolution of the PSNR and the energies with respect to the size of the search space. Here, the search space is a 2D disk and the sizes indicated on the plots correspond to the radius of this disk.

source image are automatically rejected and need to be resampled again. For this denoising example, the motion of the objects in the scene where rather small. Let r be the maximum motion in pixels of all the pixels of the scene. Of course, r is unknown as there is no ground truth available. When looking at the results in figure 3.19, we can make several observations. First, it is clear that the PSNR and the energies get better as the range shrinks from 1000, which indicates that targeting the search space is important to the final quality of the result. Then, we also observe a sharp drop in PSNR and a sharp increase in the energy for values smaller than 25. This can be explained by the fact that r must be around 25, and using a smaller radius actually stops the optimisation from finding the optimal values for the displacements. In other words, the search space must not be too small so that it contains all the optimal values, but should not be too large either otherwise it is more difficult for the algorithm to find those optimal values. Therefore, we conclude that one would benefit from incorporating prior knowledge about the problem if available and setting constraints on the search space.

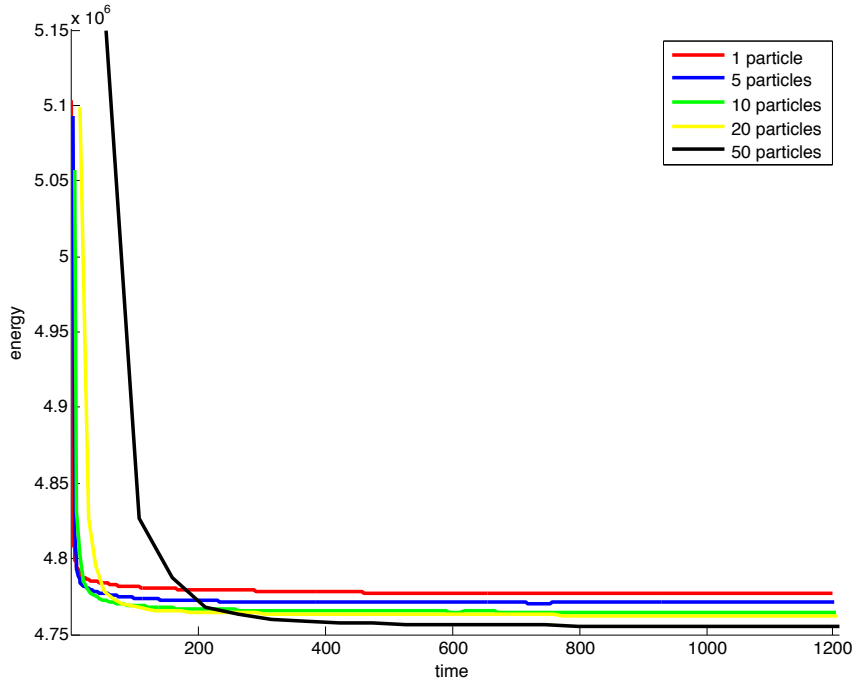
Varying the Number of Particles

The particle count is an important parameter of the algorithm. Here we run the denoising application of section 3.7.2 with different number of particles and plot the energy for each case. Results can be seen in figure 3.20. We notice that using less particles tends to reach a lower energy at first. However, as time increases, having more particles is beneficial and allows the algorithm to reach a lower energy later on. As mentioned in section 3.2.3, the complexity of the message computation is quadratic with respect to the number of particles when a simple array is used to store the particles. This contributes to the slower behaviour of the algorithm when having more particles.

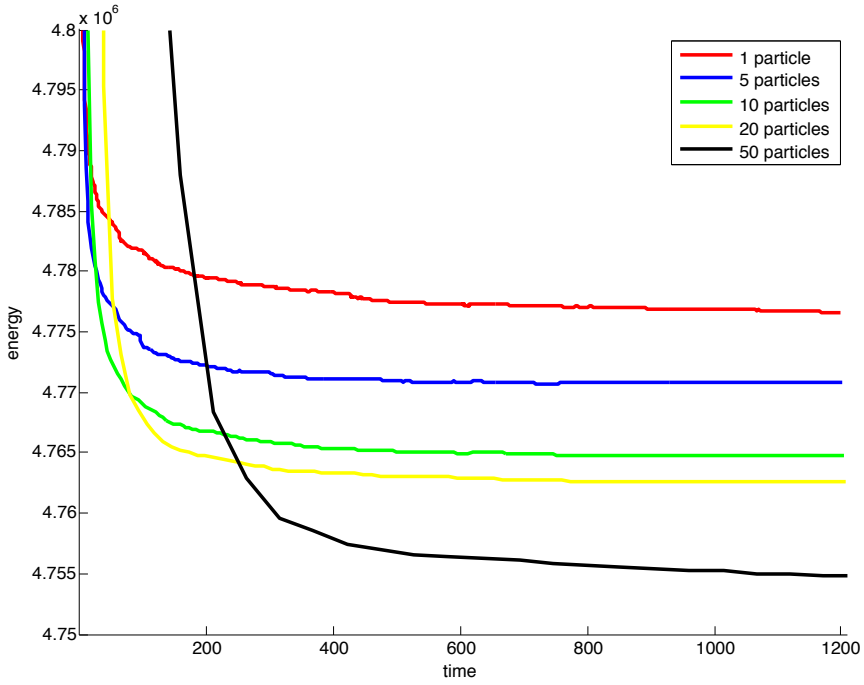
Limitations

We have demonstrated the benefits of PMBP over both PM and PBP in term on optimising in a large continuous space, however the algorithm exhibits some weaknesses under certain conditions.

PMBP is very efficient even when using a low numbers of particles (eg. 1-20), while PBP requires more particles to produce reasonable results. We can relate the density of the particle set with the level of approximation that is being performed on messages and beliefs, where higher number of particles

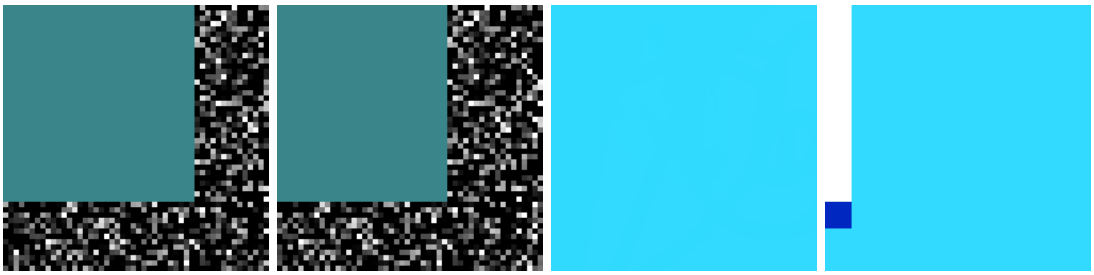


(a) Evolution of the energy

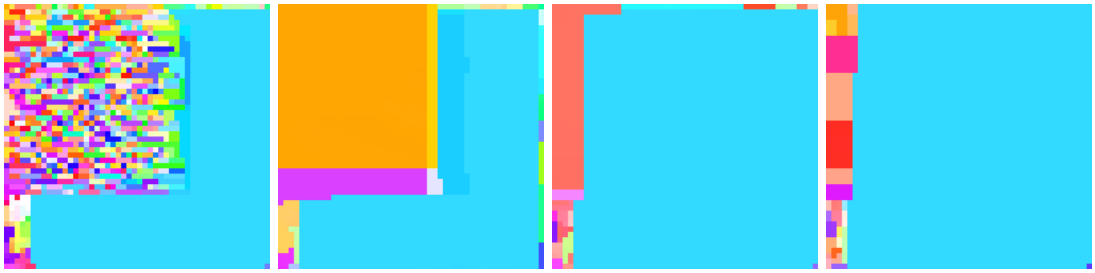


(b) Evolution of the energy zoomed on the range $[4.75, 4.8] \times 10^6$

Figure 3.20: Comparison of the energies produced by the algorithm run with different particle counts.

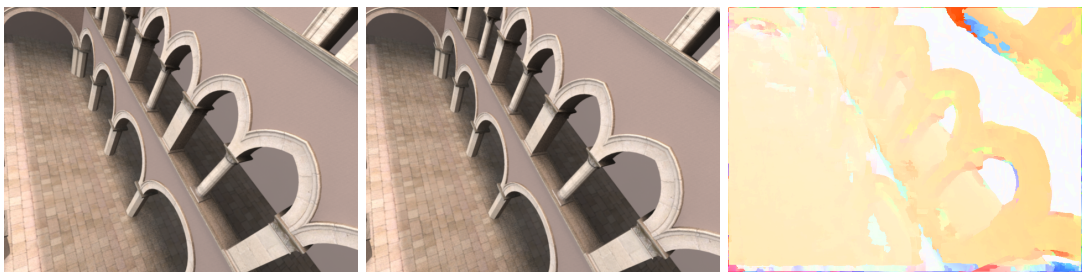


(a) Target image (b) Source image (c) Ground truth motion field (d) BP solution



(e) PM solution (f) PMBP solution with 1 particle (g) PMBP solution with 10 particles (h) PMBP solution with 25 particles

Figure 3.21: Experiment setup that illustrates the limitations of PMBP with low numbers of particles.



(a) Image 1 (b) Image 2 (c) Motion Field

Figure 3.22: Limitations of PMBP illustrated by the 2D-PCE application on flat surfaces. We can see that the textureless area above the arches produces a uniform but wrong solution, as the flow should be in agreement with its surroundings.

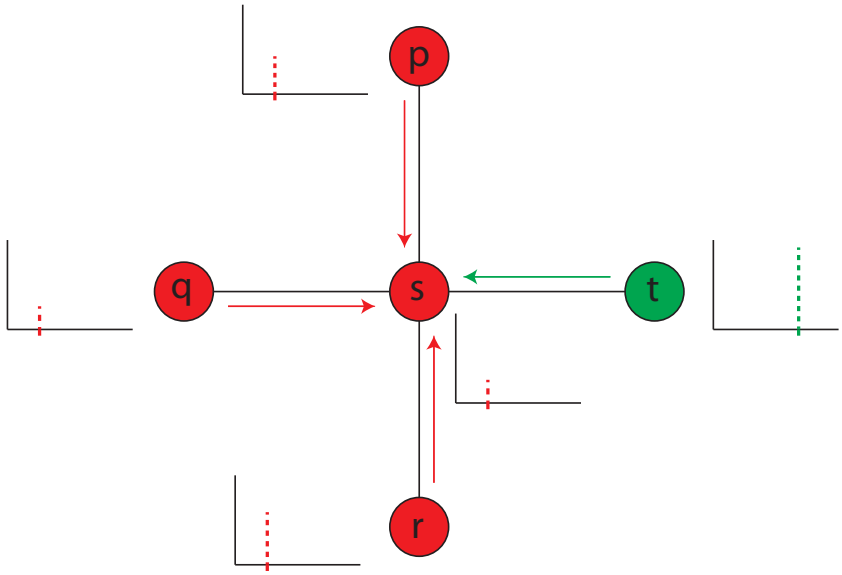


Figure 3.23: Illustration of the problem when using low number of particles. Here we suppose that the red nodes correspond to pixel in a textureless area, and the green node corresponds to a node in a textured area. The three red nodes have a strong consistent belief at a wrong particle position. The green node also has a strong belief at a particle position that is correct, as its texture is distinctive enough to be easily matched to the other image. The problem is that when s uses the particle position of its neighbour t as candidate, there will be a strong message from t to s while evaluating the new belief at s. However, the pairwise evaluation at this particle between the three other nodes (p, q and r) will yields a low belief as the existing particles of p, q and r can be far from the new candidate particle coming from t. This is another type of pairwise-miss that happens under these conditions. When using more particles, the particles positions at p, q and r are more spread and will allow for a more correct message evaluations and the particle at t will be propagated successfully to the set of s, and it turn p, q and r.

produces a better approximation and vice-versa. However when using a very low number of particles, such as 1, the algorithm struggles at propagating information into areas where the data term is not distinctive enough. This behaviour can be seen in figure 3.21. The image here is separated into two distinct areas, a textured one, and a plain one. The displacement is ambiguous in the plain area, however BP should naturally propagate the displacement of the textured areas into the plain region, as seen in the same figure. PM produces a noisy displacement field, and PMBP generates a smooth, but wrong solution in this area. This behaviour can also be seen on a non trivial case in figure 3.22 with the 2D-PCE application.

To illustrate what triggers this behaviour, we consider the case shown in figure 3.23 of the nodes at the textured/plain area boundary. The middle node is the node that is currently being resampled, and is connected to 1 textured pixel and 3 plain pixels. Here, the textureless area has a strong consistent solution, however the solution of the textured pixel should be the correct one. With the propagation step of PatchMatch, we use the textured pixel to get particle samples. The problem is that while the message

from the texture node to the middle node has a very low disbelief at this new particle, the "pairwise-miss" phenomenon is occurring at the messages coming for the plain nodes. Because they only have 1 particle in their particle sets, if this particle is spatially far from the new proposed particle, the pairwise term will produce a high disbelief. When using more particles, it is more likely that the message coming from the other nodes will suffer less from pairwise-misses, and the new particle can then be successfully accepted into the particle set of the middle node.

3.8 Discussion

In this chapter we have established a fundamental link between two important algorithms: PatchMatch and Belief Propagation, by expressing both algorithms into a common framework and analysing their similarities. We used this analysis to design a new algorithm, PatchMatch Belief Propagation (PMBP), which combines PM and BP. This new algorithm is able to overcome the weaknesses of PM, by including an explicit regularisation term to the optimisation, and solve the performance problem linked to BP by accelerating it by several orders of magnitudes.

We expressed three algorithms: PM, BP and PMBP into a common framework which can be seen in table 3.1. This table can be used to define the implementation of these algorithms step by step and explicitly indicates their similarities and differences. We made publicly available our new method by building an open source implementation, which is easily extendable to a various range of applications. It is available on GitHub at the following address: https://github.com/fbesse/pmbp. We also pointed out several important implementation considerations in order to fully and accurately describe the fundamental mechanisms of the algorithm, concerning the caching mechanisms, particle update, message foundation normalisation and early termination.

We conducted series of experiments showing the differences, advantages, limitations and performances of the different algorithms. From the outcome of these experiments we can draw several important conclusions. First, PM is a very efficient algorithm to solve optimisation problems when only a unary energy is required. However, on applications where regularisation of the solution is required, PM shows its limitation. Belief Propagation algorithms are designed to address these limitations. We conducted experiments on toy examples to illustrate this.

On the other hand, there are some inherent problems linked with BP due to the fact that it can only run on a discretised state space, and exhibits poor performance on larger state spaces, or when using finer levels discretisation. Particle Belief Propagation can be used to solve these two problems on some level, however the performances are still order of magnitude slower than what would be needed to produce results in a reasonable time. We carried out several experiments to show the benefits of PMBP. While succeeding on toy experiments where all the other algorithms showed difficulties, it can also be successfully used on real-world application involving a large number of data, showing an actual potential for imaging applications. We analysed the different steps of the algorithm and discussed their importance to PMBP, and showed the difference in behaviour when modifying crucial parameters, such as the particle count and the size of the search space. An advantage of PMBP is that it is easy to incorporate prior knowledge of the problem into the optimisation by setting constraints on the search

space, we have seen are beneficial to the quality of the solution.

Finally, we showed the main weakness of PMBP which can happen when opting for a very aggressive approximation using a very small number of particles. While the performance with such a low number of particles is significantly better than PBP, the algorithm fails to successfully propagate information into textureless areas.

Algorithms such as PBP, NPB or PMBP are fundamentally trying to find the best approximation of beliefs and messages with a budget of some given objects. For PBP and PMBP, those are particles, while for NBP they are Gaussians. The goal of these algorithms is to approximate as accurately as possible the distributions that form beliefs and messages. In our case, we show that using a very low number of particles, which can be as low as 1 is often enough to produce acceptable results. However, there are some important trade-offs when using such rough approximations. We showed in the experiments that using only 1 particle makes it difficult to carry information that would allow for an accurate propagation in areas where the unary term is not distinctive enough. Future work includes refining the message and belief model used in PMBP, in order to improve the behaviour with low numbers of particles.

The next chapters of this thesis illustrate the use of PMBP for stereo matching and optical flow.

Chapter 4

Stereo matching

4.1 Introduction

Classic stereo methods often use a 1D parametrisation to estimate disparities, including pairwise terms in the optimisation in order to try to obtain a smooth disparity field, which encourages groups of adjacent pixels to have a constant disparity. However, this does not reflect the true underlying geometry of the scene as for slanted surfaces the disparity field varies smoothly rather than being constant. This affects the model in two ways. First, this assumption is often made across all the pixels of a patch. This does not hold for two reasons: the pixels of a patch might belong to different surfaces, and the surfaces might be slanted, i.e. the pixels do not have the same disparity (or depth). This affects the data (or unary) term. Second, the pairwise term is also affected as varying disparities across several pixels are usually penalised. In the case of slanted planes, this penalty is unjustified.

Some methods such as [Woodford et al., 2009], propose the use of the second order smoothness prior which encourages the *derivative* of the disparity field to be constant (and thus the *second derivative* to be null), which addresses this problem at the expense of using a more complex pairwise model. Recently, [Bleyer et al., 2011] proposed a different approach, "PatchMatch Stereo", which consists in over-parametrising the disparity field to reflect the presence of individual 3D planes at each pixel, and make the assumption that the scene consists of several planar surfaces. This has the advantage of improving the standard 1D-model for one that is more geometrically correct. Yet, as the optimisation algorithm used in their approach is PatchMatch, the authors do not explicitly use a pairwise term in their optimisation, and only perform a final post-processing step for smoothing out undesirable disparities. We propose to use PMBP with the PatchMatch Stereo parametrisation to address PatchMatch's problems and explicitly introduce smoothness in the solution. We design a pairwise term providing smoothness to the field of 3D planes that the algorithm optimises, and we show that we obtain better performance than PatchMatch stereo, and state-of-the-art results on the Middlebury dataset [Scharstein and Szeliski, 2002].

4.2 Method

4.2.1 Model

PatchMatch stereo focusses on designing a strong unary term, which can then be used within the PatchMatch algorithm. Our method is based on PatchMatch stereo and we use the same unary term. In this section we explain in detail our algorithm using the notation of Chapter 3. First, it is important to note we do not have any calibration information, and thus we operate in disparity space, instead of the regular physical space. That is, the third coordinate of a point p, which we call z_p (sometimes also called d_p), is the *disparity* value of the point. The algorithm aims at estimating a plane at each pixel, which is a plane in disparity space. Moreover, a plane in disparity space is also a plane in physical space [Trucco et al., 2003] when using rectified images, and therefore our assumption that the scene is made of planar surfaces, in geometry space still holds with this formulation.

At each pixel s, a state $\mathbf{u}_s$ represents the equation of a plane $f_{\mathbf{u}_s}$, such that $\mathbf{u}_s = [\mathbf{n}, z]$, where $\mathbf{n}$ is the normal of the supporting plane and z is its disparity. As the normal of the plane is a normalised value, we only need to store its x and y components, such that $\mathbf{u}_s = [n_x, n_y, z_s]$ with $n_z = \sqrt{1 - n_x^2 - n_y^2}$. We make the general assumption that all pixels of a patch are located on the same plane. At a pixel t located on a patch centred around s, we can compute the following disparity:

$$z_t = a_s x_t + b_s y_t + c_s \tag{4.1}$$

where $a_{\mathbf{u}_s}$, $b_{\mathbf{u}_s}$ and $c_{\mathbf{u}_s}$ are precomputed values:

$$\begin{aligned} a_s &= -\frac{n_x}{n_z} \\ b_s &= -\frac{n_y}{n_z} \\ c_s &= \frac{n_x x_s + n_y y_s + n_z z}{n_z} \end{aligned} \tag{4.2}$$

To explain the origin of these three coefficients, we suppose that pixels s and t are located on the same plane (as they belong to the same patch), and therefore must follow the same plane equation:

$$n_x x_t + n_y y_t + n_z z_t + D = 0 \tag{4.3}$$

$$n_x x_s + n_y y_s + n_z z_s + D = 0 \tag{4.4}$$

where D is a constant. From these two equation, we can derive the following:

$$n_x x_t + n_y y_t + n_z z_t l = n_x x_s + n_y y_s + n_z z_s \tag{4.5}$$

$$\Leftrightarrow$$

$$n_z z_t = -n_x x_t - n_y y_t + n_x x_s + n_y y_s + n_z z_s \tag{4.6}$$

$$\Leftrightarrow$$

$$z_t = -\frac{n_x}{n_z} x_t - \frac{n_y}{n_z} y_t + \frac{n_x x_s + n_y y_s + n_z z_s}{n_z} \tag{4.7}$$

$$\Leftrightarrow$$

$$z_t = a_s x_t + b_s y_t + c_s \tag{4.8}$$

where $a_{\mathbf{u}_s}$, $b_{\mathbf{u}_s}$ and $c_{\mathbf{u}_s}$ are defined above.

It is useful to store the two representations mentioned before in memory. Indeed, the plane normals and disparities at each pixel $[n_x, n_y, z_s]$ are useful during the randomisation step, as explained below, and the precomputed values $[a_s, b_s, c_s]$ can be used to evaluate in an efficient way the disparities of all the pixels of a patch.

Using this parametrisation, the unary cost ψ_s^{pms} resembles ψ_s^{wpf} defined in Equation 3.3:

$$\psi_s^{\text{pms}}([a_s, b_s, c_s]^\top) = \sum_{i=-h}^{h} \sum_{j=-h}^{h} w_{sij}\, \rho\Big(I_1\big(x_s + i,\ y_s + j\big), I_2\big(x_s + i + (a_s i + b_s j + c_s),\ y_s + j\big)\Big). \tag{4.9}$$

where the parameters of the state are displayed in red, and where w_{sij} is a weight defined as:

$$w_{sij} = \exp(- \parallel I(x_s, y_s) - I(x_s + i, y_s + j) \parallel / \omega). \tag{4.10}$$

and represents the adaptive support weight [Yoon and Kweon, 2006] for the different pixels of the patch centred at pixel s. The parameter ω controls the strength of the adaptive support weight. An illustration of the adaptive support weight computation can be seen in figure 4.1. This computation outputs weights for each pixel of a given patch, which can then be used as support when calculating the unary term, based on the colour data of the patch. Similarly to a bilateral filter, pixels that are closer and more similar in colour to the central pixel will have higher weights. The term ρ is an error measure that penalises difference in pixel colour and in pixel gradient:

$$\begin{aligned}\rho(I_1(x,y), I_2(x',y')) = (1-\alpha)\min(\|I_1(x,y) - I_2(x',y')\|, \tau_{\text{col}}) + \\ \alpha \min(\|\nabla I_1(x,y) - \nabla I_2(x',y')\|, \tau_{\text{grad}})\end{aligned} \tag{4.11}$$

where α is a parameter that balances the influence of the colour and gradient term, and parameters τ_{col} and τ_{grad} are truncation terms for robustness in occluded regions. The addition of the gradient terms allows the algorithm to be more robust to difference in luminosity.

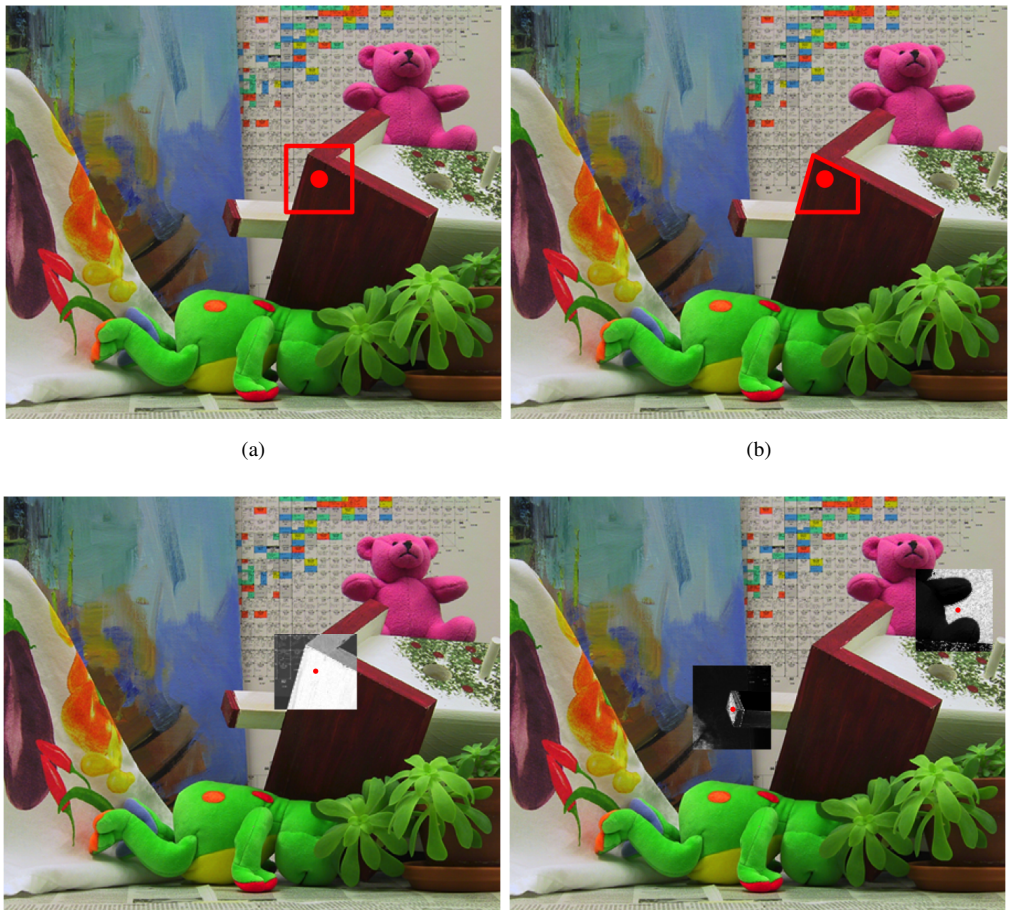
We introduce an alternative notation for the similarity measure. Let $\hat{I}(x,y)$ be the 4 dimensional vector consisting of a concatenation of the colour values and the gradient values of the image at pixel (x,y). Let θ be a 4 dimensional parameter vector defined as $\theta = [\alpha_c, \alpha_\nabla, \tau_c, \tau_\nabla]$. For any 4 dimensional vector $\hat{v} = [v_1, v_2, v_3, v_4]^\intercal$, we define the norm $\left\|\hat{v}\right\|_\theta$ as:

$$\left\|\hat{v}\right\|_\theta = \alpha_c \min\left(\|[v_1, v_2, v_3]^\intercal\|, \tau_c\right) + \alpha_\nabla \min\left(v_4, \tau_\nabla\right) \tag{4.12}$$

We can then define ρ as:

$$\rho(I_1(x,y), I_2(x',y')) = \left\|\hat{I}_1(x,y) - \hat{I}_2(x',y')\right\|_\theta \tag{4.13}$$

with $\theta = [1-\alpha, \alpha, \tau_c, \tau_\nabla]$.



(a) (b)

(c) (d)

Figure 4.1: Illustration of the adaptive support weights computation. (a) shows a patch whose middle pixel lies on a certain surface - note that the patch contains other pixels that are not on the same surface. (b) shows the pixels of the patch that are located on the same surface as the middle pixel. (c) shows the weights of the adaptive support computation, where white is 1 and black is 0. (d) shows adaptive support weights for two other patches.

4.2.2 Optimisation

PatchMatch stereo uses PatchMatch to optimises the unary energy ψ_s^{pms} defined previously. As mentioned in Chapter 3, there are three crucial steps that we need to describe: random initialisation, propagation and randomisation. In addition to that, the authors define a new step called "view propagation" which is also explained below.

Random Initialisation This step consists in initialising each state with a random plane. While it is possible to sample the values $[a_s, b_s, c_s]$ randomly, it is not clear whether this will cover the space of all possible planes. Instead, the plane normal and disparity parametrisation is used. First, the disparity is sampled uniformly within the range of possible values. Then, the plane normal is uniformly sampled within the unit hemisphere facing the camera. These values are then converted into the more convenient $[a_s, b_s, c_s]$ parametrisation. Note that the disparities are sampled with sub-pixel accuracy.

Propagation In this step, the current best plane defined by state $\mathbf{u}_t$ of pixel t is proposed as candidate to a neighbouring pixel s. However, there is a conversion that needs to take place as the disparity at s given by this plane is different from that of t. Assuming that:

$$\mathbf{u}_t = [n_x, n_y, z], \tag{4.14}$$

we define the candidate state $\mathbf{u}_s$ as:

$$\mathbf{u}_s = [n_x, n_y, z'] \tag{4.15}$$

where

$$z' = -\frac{n_x}{n_z}x_s - \frac{n_y}{n_z}y_s + \frac{n_x x_t + n_y y_t + n_z z}{n_z}. \tag{4.16}$$

The value z' is derived simply by assuming that pixels $s = [x_s, y_s, z]$ and pixel $t = [x_t, y_t, z']$ are on the same plane, and thus follow the same plane equation. This adjustment was not incorporated in [Bleyer et al., 2011].

Randomisation The randomisation step consists in sampling a new random plane at a given distance from a given existing plane. To do so, we use again the parametrisation defined by the plane normal and the disparity as it is convenient to jitter these values rather than the coefficients a, b and c. We compute the differential values Δ_z and Δ_n which are sampled from the intervals $[-\Delta_z^{\max}, \Delta_z^{\max}]$ and $[-\Delta_n^{\max}, \Delta_n^{\max}]$ respectively. The sizes of these intervals varies according to the standard concentric circle sampling mechanism of PatchMatch. After having retrieved these random differential values, we compute the final candidate $u'_s = [\mathbf{n}', z']$ with $\mathbf{n}' = \mathbf{n} + \Delta_n$ and $z' = z + \Delta_z$.

View Propagation In addition to the previous three standard steps, PatchMatch stereo benefits from an extra type of propagation if the matching is done in a bidirectional way (left image matched to the right image and inversely). Indeed, the forward and backward solution are largely related. In terms of 2D parametrisation, if the offset at pixel s in the left view is $[dx_s, 0]$, then the offset of the pixel $[x_s + dx_s, y_s]$ in the right view is $[-dx_s, 0]$. In terms of the parameterisation used in PatchMatch stereo, due to the fact that it uses a geometric representation, the states of two matching pixels are equivalent (same plane normals, opposite disparities) as they represent the same physical plane. Only the disparity value needs to be adjusted to account for the direction (left to right or right to left) of the matching. Thus, for each view, the states of each pixel are stored somewhere in memory, and used afterwards as candidate states for the matching pixels. Figure 4.2 illustrates this view propagation process, also called mirroring.

4.2.3 Post-Processing

After retrieving the solution computed by PatchMatch, a post-processing step takes place to improve it, and use the method employed in PatchMatch stereo. First, we note that the solution is computed in a bidirectional way, and thus the two solutions (left to right and right to left) are available. A left/right consistency checking test is then run on both solutions. This is a non-expensive way of isolating pixels that are likely to either have a wrong solution, or be occluded in either view. To do so, we can consider a pixel s in the left view, and its matching pixel s' in the right view. We compute the disparities d_s and $d_{s'}$ of those two pixels according to the two solution fields, and those pixels are flagged as consistent if and

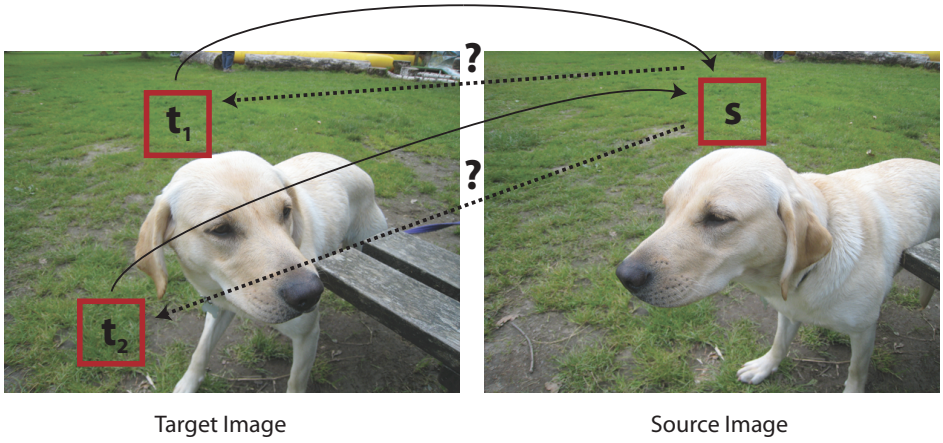


Figure 4.2: Illustration of the view propagation process. When running the matching process in the first direction, patches t_1 and t_2 both match to patch s. These matches are stored in memory, so that when the matching process is run in the inverse direction, both patches t_1 and t_2 will be proposed as candidate for s.

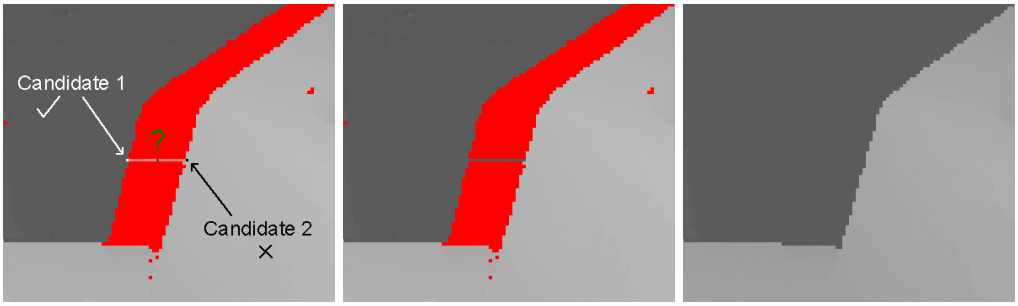


Figure 4.3: (a) shows in red the occluded pixels that need post processing. In light grey we show a segment that needs to be inpainted, and in green a pixel on this segment. We look for the first valid pixels on the left and right of this green pixel, which are shown by the white and the black pixel. After comparing their disparity, we find that the white pixel has a smaller disparity, meaning a greater depth, and thus we use its state to inpaint the whole scanline. The result after this is shown in (b). (c) shows the result after all the invalid pixels have been post-processed.

only if $|d_s - d_{s'}| < 1$. Then, an inpainting step takes place in order to correct the disparity values as the non-consistent pixels only. We assume that non consistent pixels correspond to occluded pixels. In the stereo matching application, there is no motion of the scene as both images where captured at the same moment in time, and therefore it is a valid assumption that occluded pixels have a higher depth (lower disparity) than non occluded pixels, as they are in the background in relation to the occluder. Then, for each occluded pixel, we search for the first non occluded pixels on a scanline both on its left and right sides and retrieve these solutions, which correspond to two planes u_{left} and u_{right}. As it is known that the occluded pixel belong to the background, the solution yielding the lowest disparity is attributed to the

occluded pixel. Figure 4.13 illustrates this. Following the inpainting of occluded pixels, a median filter is run on the solution field to reduce the horizontal streaks that appear due to this scanline inpainting mechanism as in [Bleyer et al., 2011].

4.2.4 Pairwise term

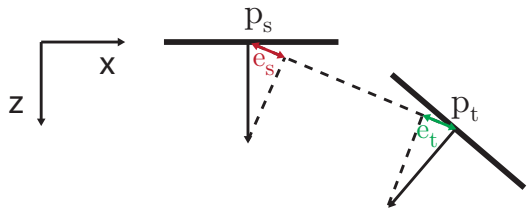


Figure 4.4: View from above (towards the y axis). p_s and p_t are two neighbouring points. Evaluation of the pairwise term : the sum of the projection of each normals onto the vector joining the two points (which are the sum of the length of the green and the red segments).

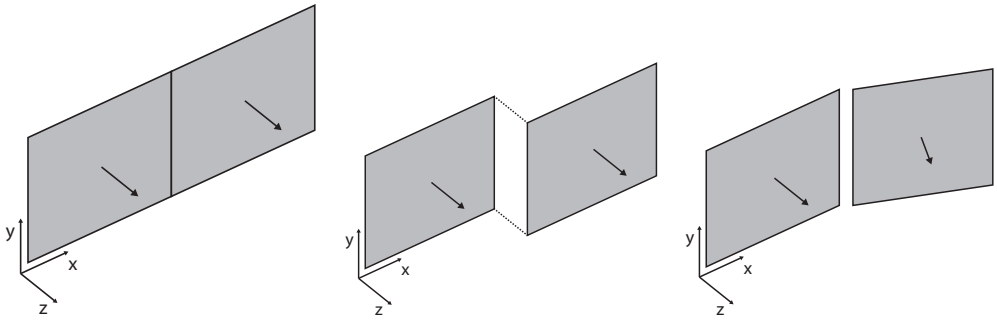


Figure 4.5: Illustration of the behaviour of the pairwise term on different cases: (a) Neighbouring planes have the same normals and the centre points are at the same depth: no penalty. (b) Neighbouring planes have the same normals but the centre point are not at the same depth: penalty. (c) Neighbouring planes do not have the same normals: penalty.

We base our method on the PatchMatch stereo algorithm, and augment it with a pairwise term that encourages continuity of the underlying geometry of the scene. Constraining the normals of two neighbouring planes to be the same is not sufficient, as they can still be located at different disparities (or depth in physical space). Instead, we use the projection of each point onto its neighbouring plane as a dissimilarity measure, that is defined as:

$$\psi_{st}(\mathbf{u}_s, \mathbf{u}_t) = w_{st}\big(|\mathbf{n}_s \cdot (\mathbf{x}_t - \mathbf{x}_s)| + |\mathbf{n}_t \cdot (\mathbf{x}_s - \mathbf{x}_t)|\big). \tag{4.17}$$

where the term w_{st} is once again the adaptive support weight and modulates the effect of the pairwise term according to the apparent gradient of the image, and is defined as:

$$w_{st} = \exp(- \parallel I(x_s, y_s) - I(x_t, y_t) \parallel /\omega). \tag{4.18}$$

Note that our pairwise term defined in equation 4.17 is included within the energy of equation 3.1 and the term λ controls its influence over the unary term. Our pairwise term encourages the supporting planes of two neighbouring pixels to have both the same orientation and the same depth (at the 3D points corresponding to these pixels). Setting $\lambda = 0$ gives us exactly PatchMatch stereo. An illustration of the geometrical meaning of the pairwise term can be seen in figure 4.4, while an illustration of different cases is available in figure 4.5.

4.3 Experiments

We conduct a series of experiments to show the benefits of our method. We first show the influence of using different levels of regularisation on the results using the pairwise terms that we designed. We then apply our method on the Middlebury Stereo dataset and show that we obtain state of the art results. Finally, we demonstrate the benefits of the view propagation and post-processing step, and discuss limitations and future work.

4.3.1 Varying the regularisation level

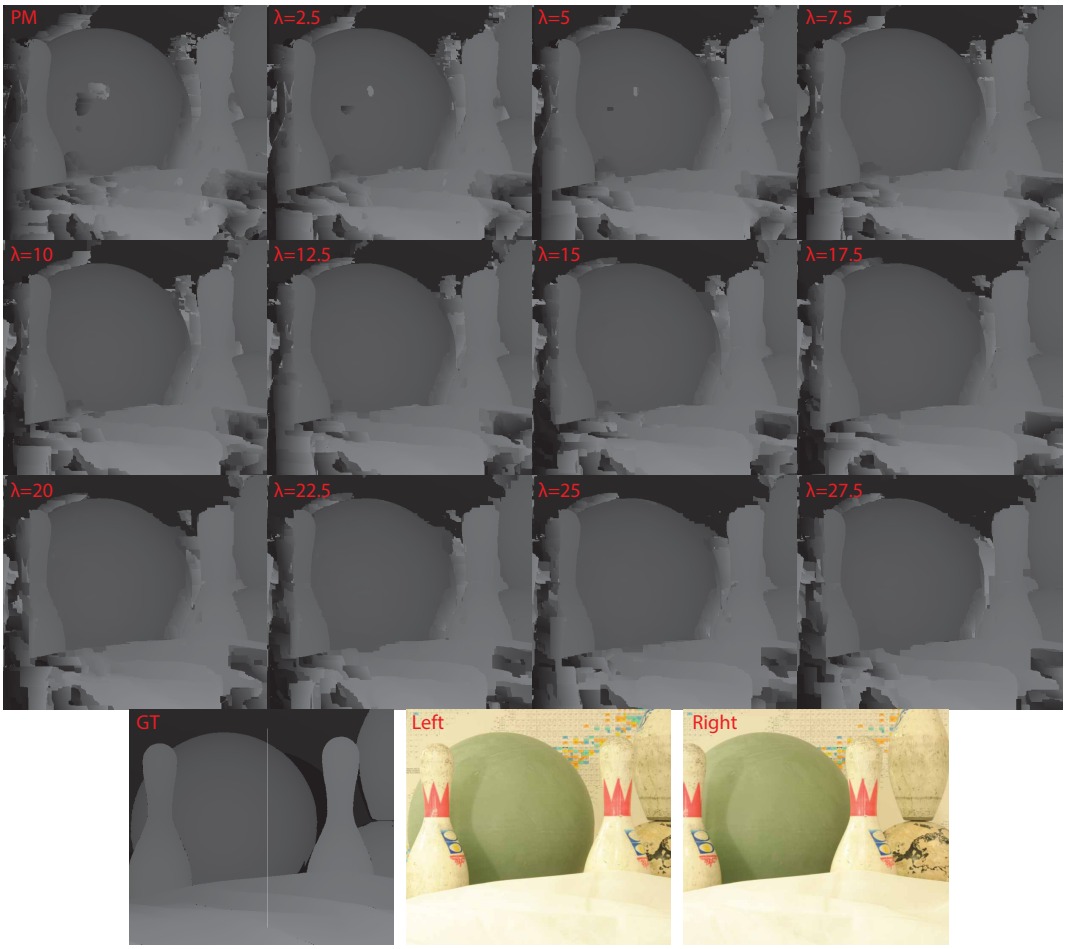
In this experiment we use four of the stereo images pairs from the Middlebury dataset, with available ground truth. We run PMBP with the pairwise term presented in equation 4.17. We vary the parameter λ of equation 3.1 which controls the level of regularisation (we recall that a value of $\lambda = 0$ corresponds in our case to the original PatchMatch stereo). We run PMBP stereo on four cases and the results are visualised in figures 4.6, 4.7, 4.8 and 4.9. The difference between the result of PatchMatch and PMBP with $\lambda = 12.5$ can be seen in figure 4.10.

On these figures we plot the error with respect to the regularisation level. Note that we display the error for both raw and post-processed outputs, to exclude the possibility that any increase in quality is a byproduct of the post-processing. On both curves for the four cases, we observe that the regularisation level that gives the best error is a value of $\lambda \neq 0$, which shows that our algorithm yields a systematic improvement over PatchMatch stereo. We also observe visually that the displacement field appears increasingly smoother with higher values of λ, as expected. An interesting area that illustrate the benefits of PMBP is the area located in the middle of the bowling ball of the case that is presented in figure 4.6. There, the texture near the centre of the bowling ball is almost textureless and not very distinctive. PatchMatch stereo fails to recover a correct disparity as can be seen on the same figure. However, by increasing the level of regularisation, we can see that this artefact gradually disappears.

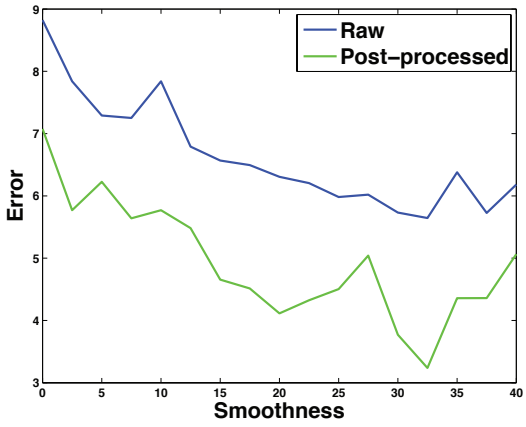
Another interesting area to analyse more in detail is the pages of the open book in figure 4.8. Similarly to the bowling ball, it contains poorly textured areas, where PatchMatch stereo produces noticeable artefacts while our method has the advantage of smoothing them out of the resulting disparity field. Naturally, we also observe an over-smoothing effect for large values of λ, which can be seen in the curves of figures 4.6, 4.7, 4.8 and 4.9.

4.3.2 Middlebury dataset

We also run PMBP stereo on the test set of the Middlebury dataset, which is shown in figure 4.11. For this, we use the same parameter as PatchMatch stereo, $\{\omega, \alpha, \tau_{col}, \tau_{grad}\} = \{10, 0.9, 10, 2\}$ with a patch



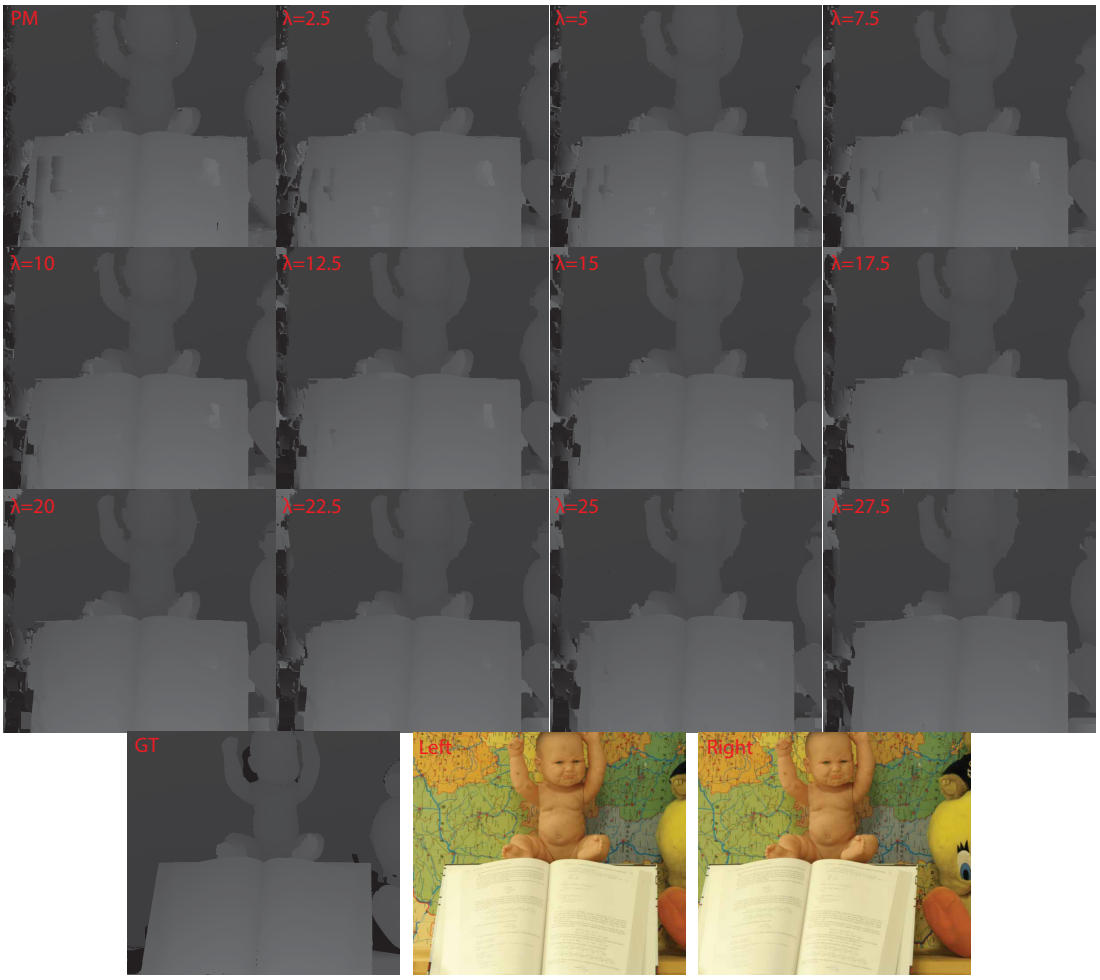
(a)



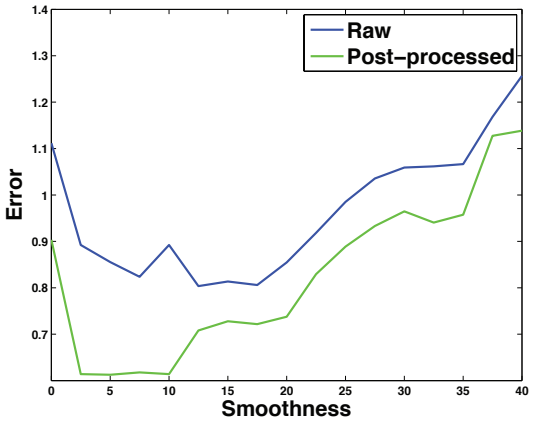
(b)

Figure 4.6: (a) Raw results (without post-processing) on the Bowling1 dataset. λ is a pairwise weighting coefficient, controlling the amount of regularisation; (b) Evolution of the error with respect to the smoothness coefficient.

size of 40x40 pixels. The weighting of the pairwise term is set to $\lambda = 7.5$, after verifying experimentally that it gives the best possible results. The results are summarised in table 4.1 and figure 4.12. We observe

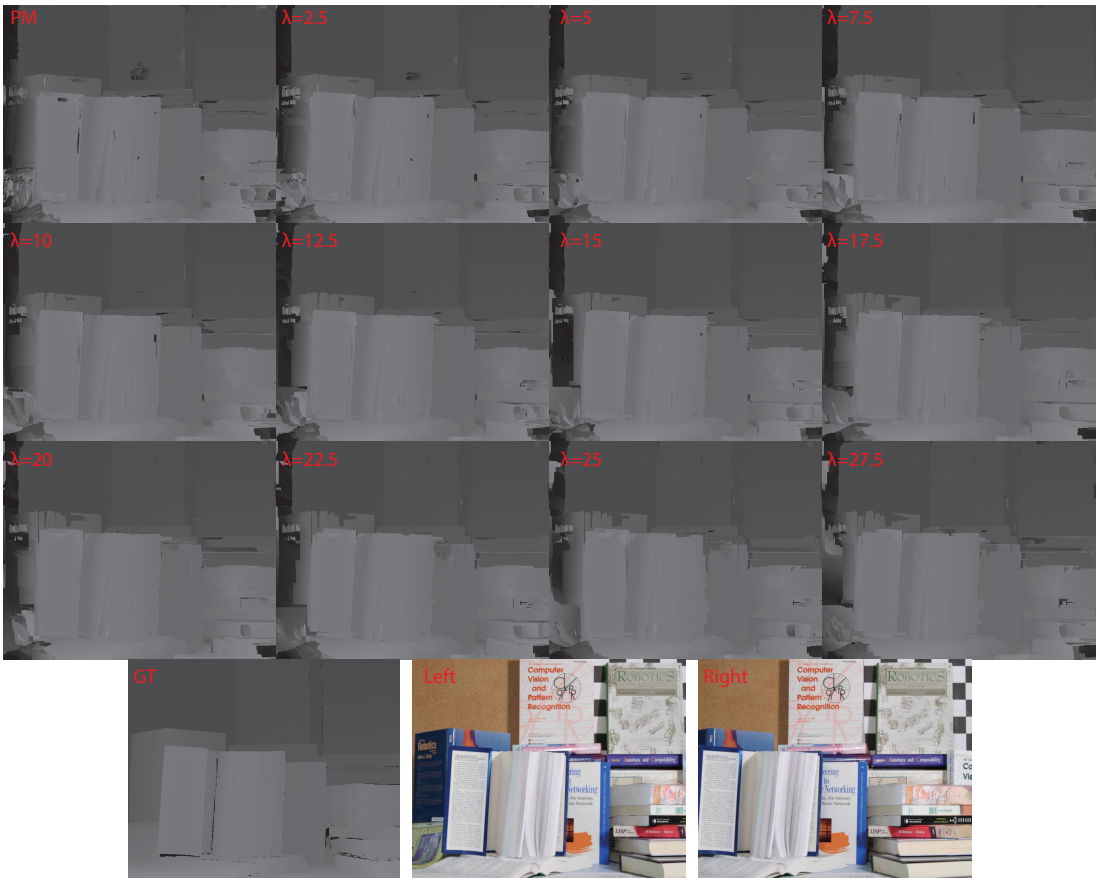


(a)

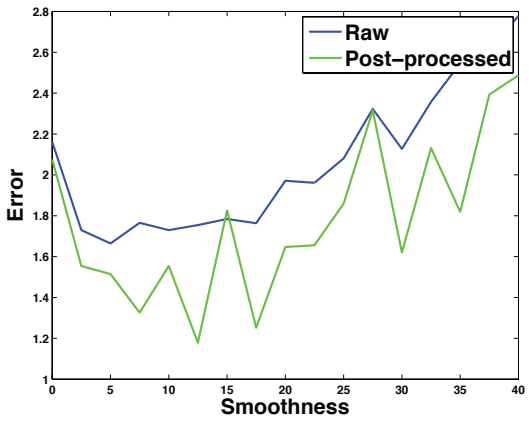


(b)

Figure 4.7: (a) Raw results (without post-processing) on the Baby2 dataset. λ is a pairwise weighting coefficient, controlling the amount of regularisation; (b) Evolution of the error with respect to the smoothness coefficient.



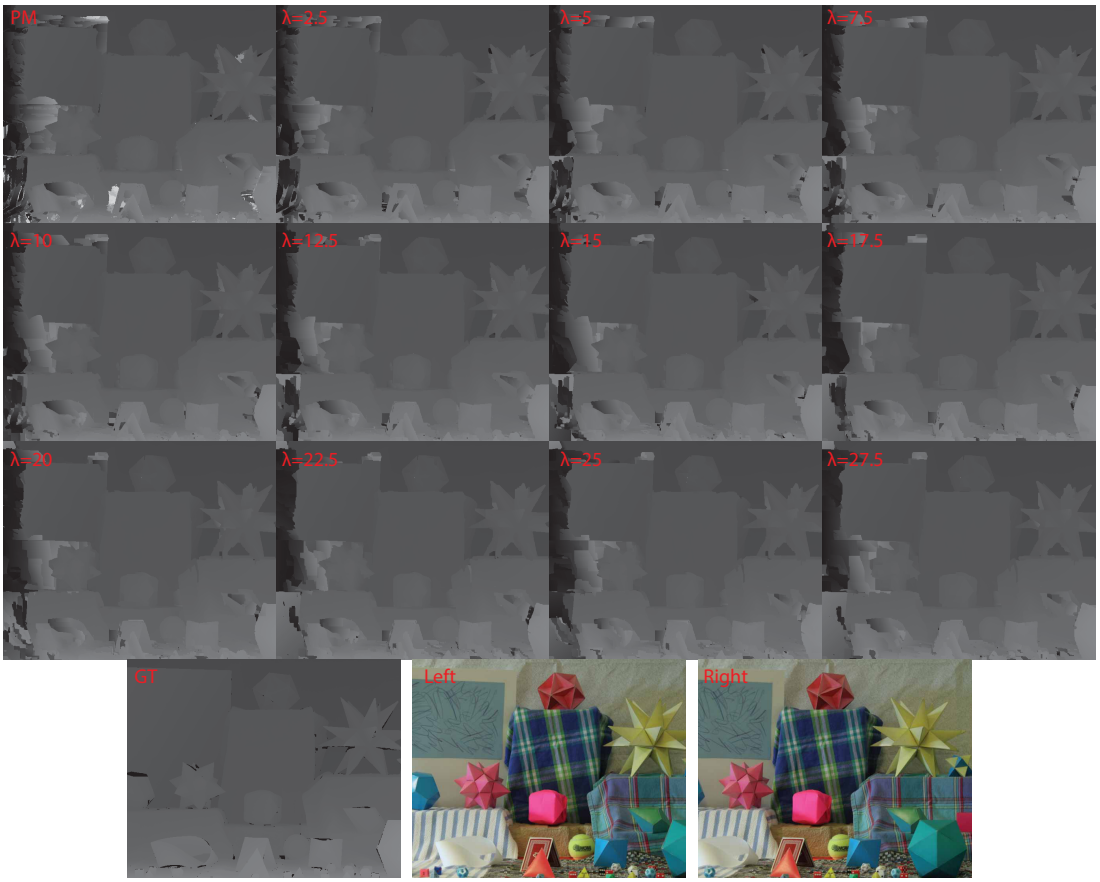
(a)



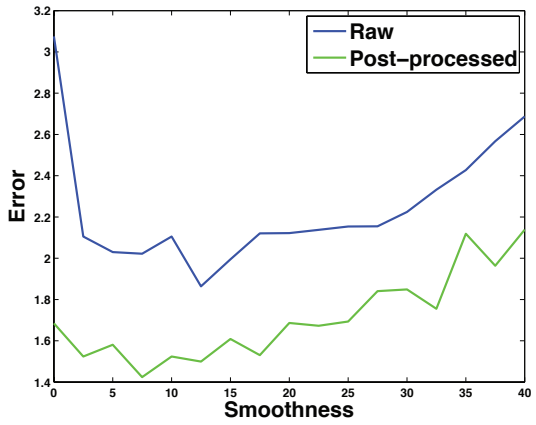
(b)

Figure 4.8: (a) Raw results (without post-processing) on the Books dataset. λ is a pairwise weighting coefficient, controlling the amount of regularisation; (b) Evolution of the error with respect to the smoothness coefficient.

that PMBP stereo is superior to PatchMatch stereo in almost all cases. For the sub-pixel accuracy level, we were overall Rank 1 of all methods at the time of publication of our paper [Besse et al., 2012]. Note that we perform particularly well on the challenging dataset “Teddy” and “Cones”.



(a)

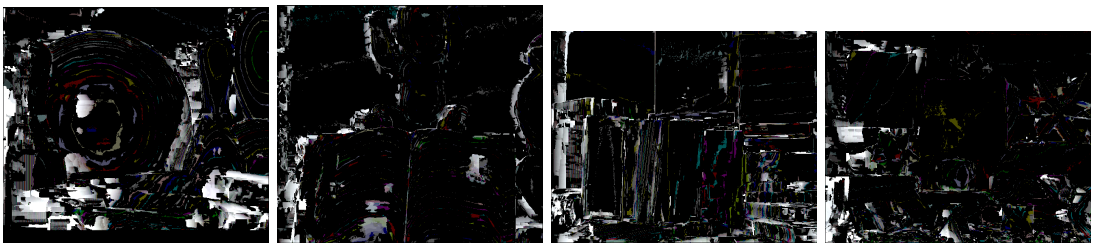


(b)

Figure 4.9: (a) Raw results (without post-processing) on the Moebius dataset. λ is a pairwise weighting coefficient, controlling the amount of regularisation; (b) Evolution of the error with respect to the smoothness coefficient.

4.3.3 Post-Processing

In this experiment, we demonstrate the effect of the post-processing step that we use after the main optimisation has finished. We use the "aloe" image pair from the Middlebury dataset. We first show the



(a) (b) (c) (d)

Figure 4.10: Graded difference of the disparity fields produced by PatchMatch and PMBP with $\lambda = 12.5$ for (a) Bowling1 dataset; (b) Baby2 Dataset; (c) Books dataset; (d) Moebius dataset.

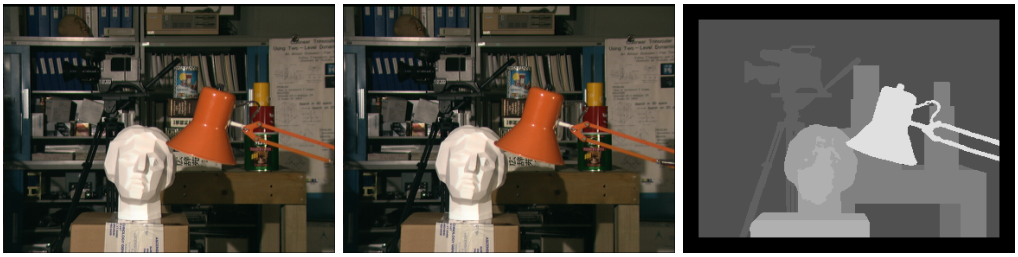
	Tsukuba			Venus			Teddy			Cones		
	nocc	all	disc	nocc	all	disc	nocc	all	disc	nocc	all	disc
PM Stereo	15.0 45	15.4 44	20.3 56	1.00 6	1.34 6	7.75 9	5.66 2	11.8 2	16.5 2	3.80 2	10.2 2	10.2 2
Ours	11.9 27	12.3 24	17.8 29	0.85 5	1.10 3	6.45 6	**5.60** 1	12.0 3	**15.5** 1	**3.48** 1	**8.88** 1	**9.41** 1

Table 4.1: Results on the Middlebury dataset with subpixel threshold (t=0.5). Bold entries indicates where our algorithm is ranked first. Our method has the first rank, with an average rank of 8.5, in contrast to 14.9 for PatchMatch Stereo.

consistency map of one test case, which consists in highlighting in white the pixels that are detected as consistent and in black those that are not. This is visualised in figure 4.13, together with the raw result and the post processed result. In this case, we used a distance threshold of 1 to isolate non consistent pixels. As this distance threshold is an important parameter, we also show a representation in grey level of this distance for each pixel, where black means null distance and white represents a distance of 2 in figure 4.16.

In general, consistent pixels are more reliable as they were matched by two different runs (forward and backward). We also show a representation of the unary and pairwise energy in the same figure. We observe that while it is clear form the consistency distance map that certain boundary pixels are occluded, this does not appear on either of the energy representation. This is due to the fact that we do not explicitly model the occlusion mechanism, and therefore those pixels find a good match - although wrong, somewhere in the image. We also observe that most of the boundaries present in the images are visible on the pairwise energy map, as the boundaries yield a high pairwise cost (although this is modulated by the conditional weight placed on the pairwise term).

We show additional post-processed results both for PM and PMBP on the Bowling1 and the Baby2 dataset in figure 4.15. We can see that the post-processing step smoothes out some of the artefacts produced by PM, however some still remain, and it is necessary to use PMBP to improve the results. The errors for varying different levels of λ (including $\lambda = 0$ which is the PM solution) of the post-processed results can be seen in figure 4.6, 4.7, 4.8 and 4.9..



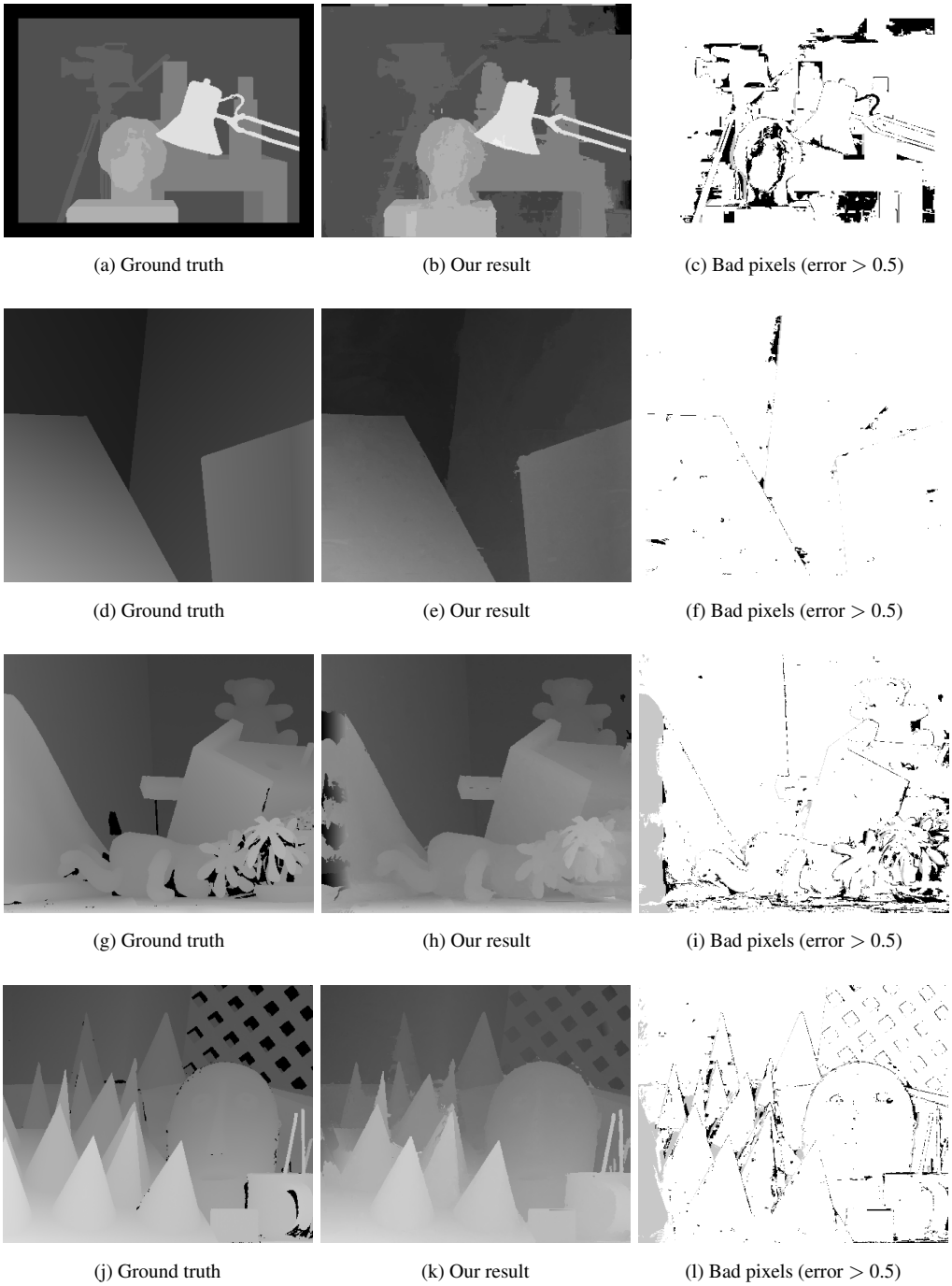
(a) Left image (b) Right image (c) Ground truth

(d) Left image (e) Right image (f) Ground truth

(g) Left image (h) Right image (i) Ground truth

(j) Left image (k) Right image (l) Ground truth

Figure 4.11: Middlebury test set. The two first columns show the left and right views respectively, while the third column is a visualisation of the ground truth disparities.



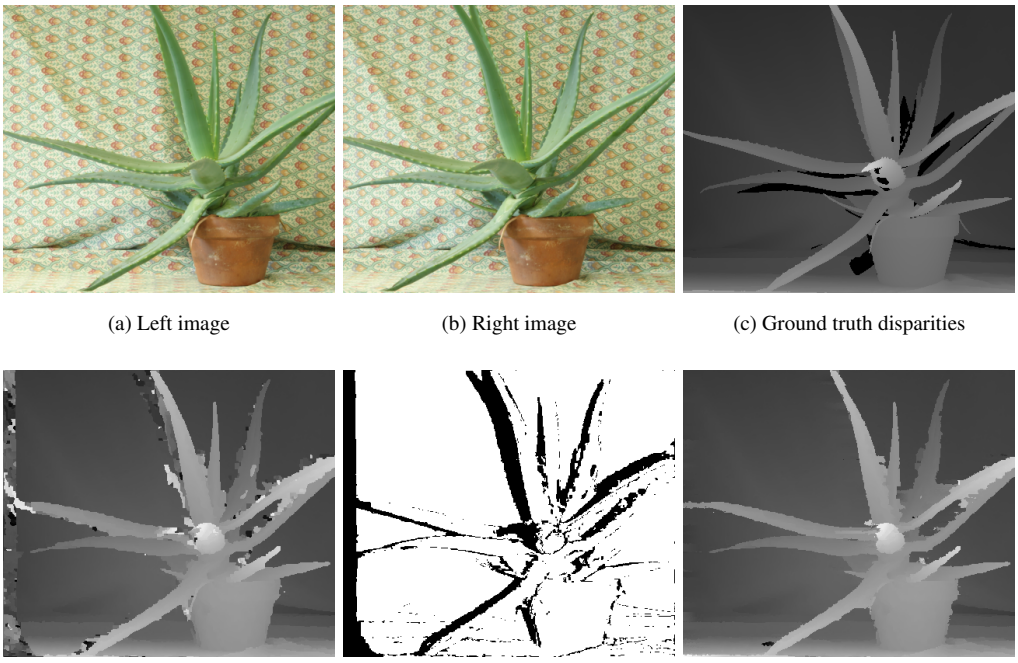
(a) Ground truth (b) Our result (c) Bad pixels (error > 0.5)

(d) Ground truth (e) Our result (f) Bad pixels (error > 0.5)

(g) Ground truth (h) Our result (i) Bad pixels (error > 0.5)

(j) Ground truth (k) Our result (l) Bad pixels (error > 0.5)

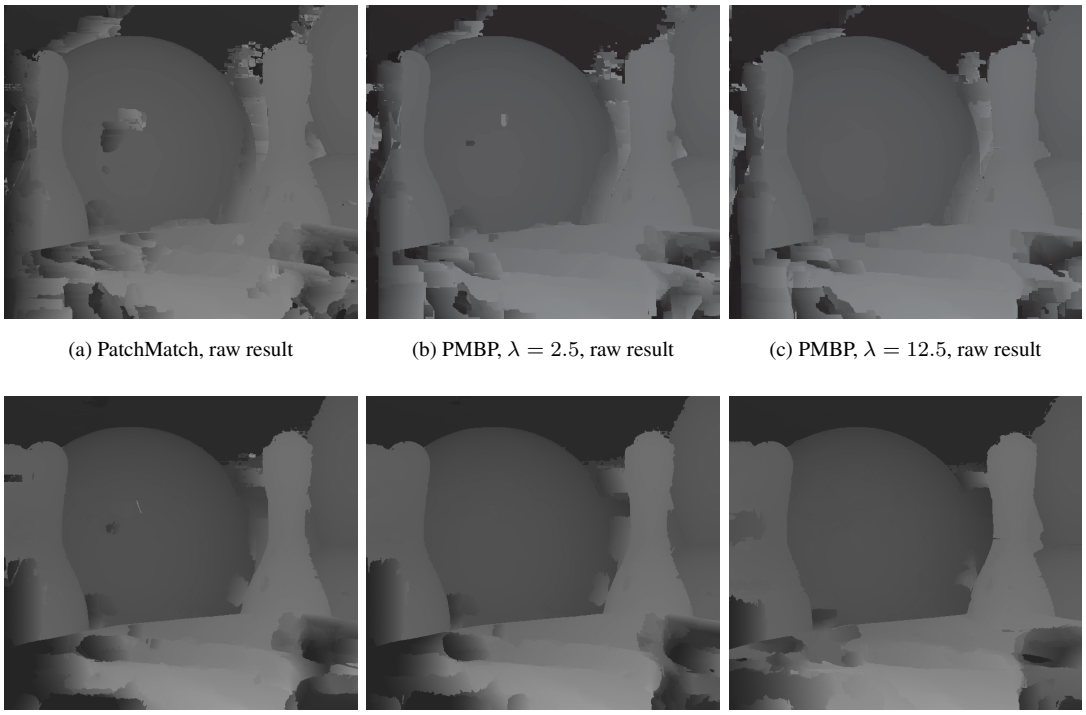
Figure 4.12: Results of PMBP on the Middlebury dataset. The first column indicates the ground truth disparity images, the second column is a visualisation of our result, and the third column represents the bad pixels, where a 'bad pixel' is a pixel whose disparity is wrong by more than 0.5 pixels.



(a) Left image (b) Right image (c) Ground truth disparities

(d) Raw result (e) Consistency map (f) Post processed result

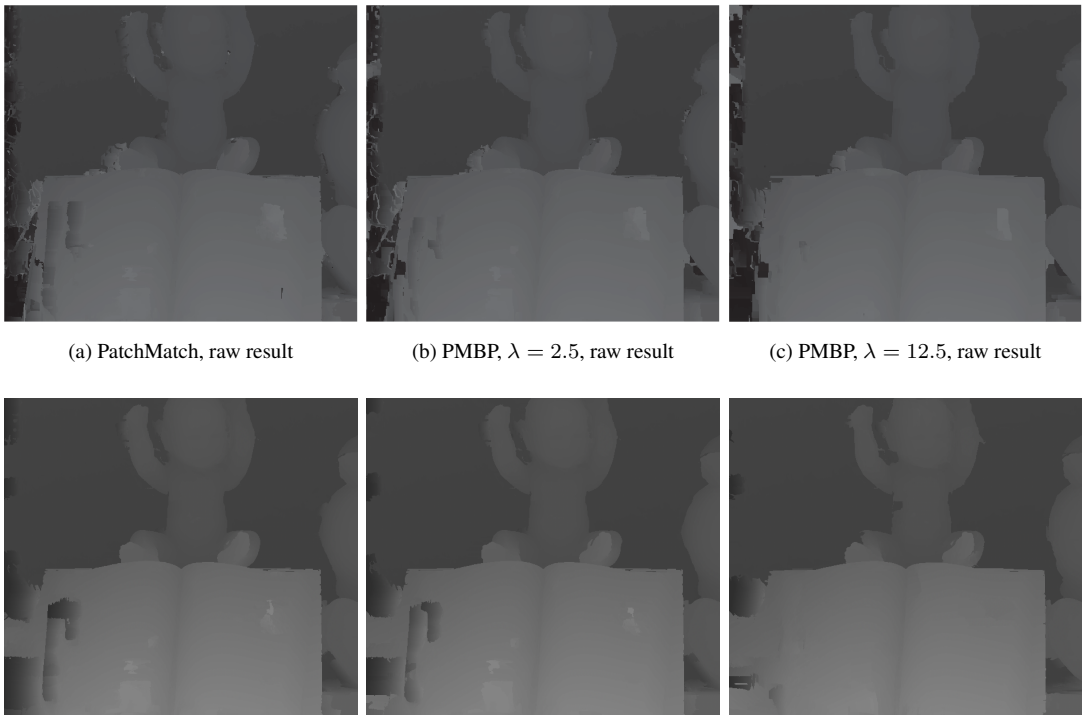
Figure 4.13: Effect of the post-processing. (a) and (b) show the two images being matches, and (c) shows the ground truth disparities. The raw result (before post processing) is shown in (d). (e) is the consistency map which is a map of valid (in white) and invalid (in black) pixels. Finally, in (f) we can see the post processed result where all the invalid pixels have been inpainted.



(a) PatchMatch, raw result (b) PMBP, $\lambda = 2.5$, raw result (c) PMBP, $\lambda = 12.5$, raw result

(d) PatchMatch, post-processed (e) PMBP, $\lambda = 2.5$, post-processed (f) PMBP, $\lambda = 12.5$, post-processed

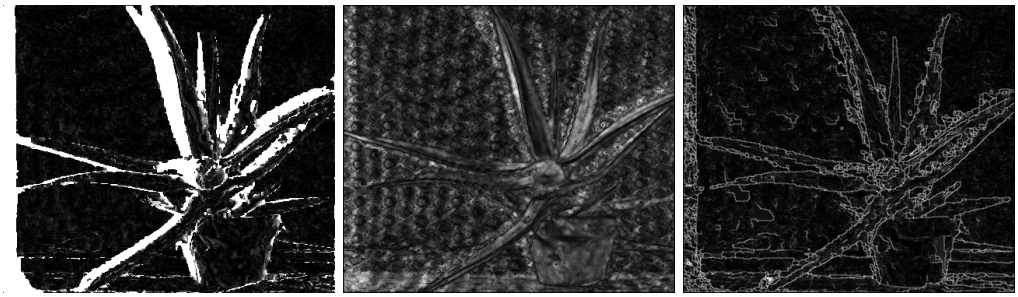
Figure 4.14: Comparison between raw and post-processed results for different settings for the Bowling1 dataset.



(a) PatchMatch, raw result (b) PMBP, $\lambda = 2.5$, raw result (c) PMBP, $\lambda = 12.5$, raw result

(d) PatchMatch, post-processed (e) PMBP, $\lambda = 2.5$, post-processed (f) PMBP, $\lambda = 12.5$, post-processed

Figure 4.15: Comparison between raw and post-processed results for different settings for the Baby2 dataset.



(a) (b) (c)

Figure 4.16: Threshold distances and energies; (a) Consistency distance. Black = 0, white = 2; (b) Unary energy. Darker values indicate lower energies; (c) Pairwise energy. Darker values indicate lower energies.

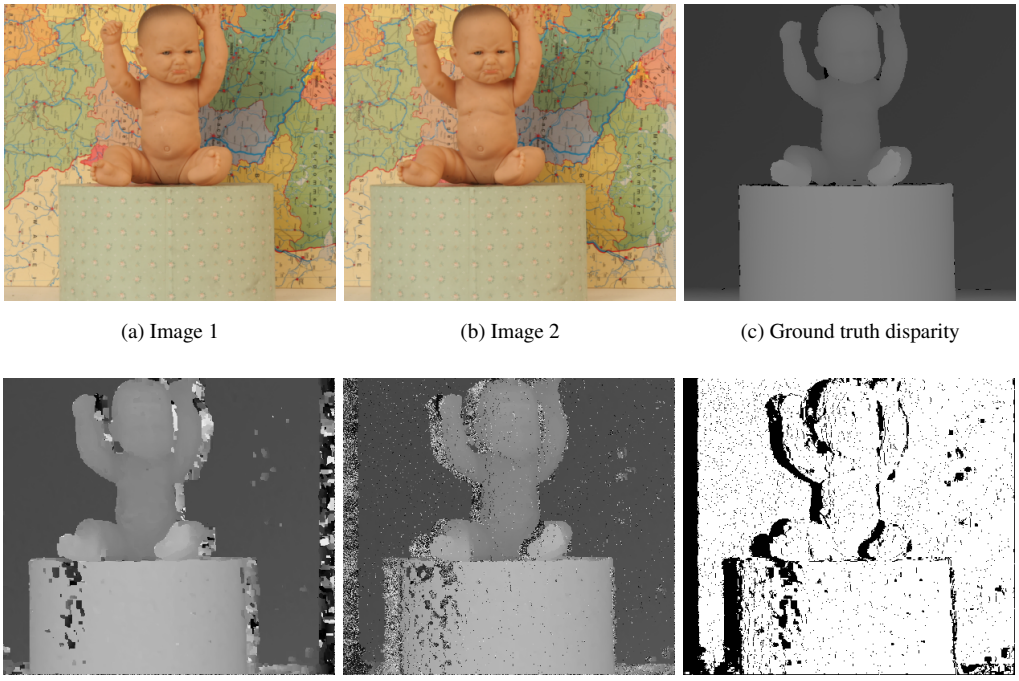
4.3.4 View Propagation

Our final experiments for this chapter consist in measuring the effects and benefits of the additional view propagation step described in section 4.2.2, we design two experiments. First we run the algorithm in a bidirectional way (left to right and right to left), however for the backward run (right to left), we disable all steps of PatchMatch except from the view propagation. This is to illustrate the effect that the view propagation has on the solution. The results can be seen on figure 4.17. We can see that most of the pixels (83%) were assigned a reasonable solution from the view propagation step only which also means that they are consistent, as seen in the related consistency map.

Lastly, we run the full PMBP algorithm with and without view propagation on one case and compare the resulting energy and error. The energy with respect to the iteration can be seen in figure 4.18. We can see that enabling the view propagation does have a positive effect on the optimisation, however the difference in energy is relatively small (view propagation improves the final energy by 0.7%).

4.4 Discussion

In this chapter we have presented an extension of the PatchMatch stereo algorithm based on PMBP. Our method allows us to integrate a pairwise term in the optimisation in order to regularise the solution, which is beneficial for the stereo matching application. Indeed, as the real-world has a piecewise continuous geometry, images of a real scene also exhibit this property, and therefore the disparity field that we are looking for is highly likely to be locally coherent. PatchMatch stereo has already proven to have good performance on the stereo application. Due to the planar representation of the scene, combined with the random initialisation and randomisation step of PatchMatch, only a few correct initial guesses per planes are needed to expand the reasonable disparity values to the whole surface. However, it is lacking the presence of regularisation, which is a defect linked to the optimiser that is used: PatchMatch, as we have shown in the previous chapter. Our algorithm also uses the mechanisms of PatchMatch, is well suited to improve PatchMatch stereo, and we are able to exactly reproduce it while benefiting from the belief propagation mechanisms when using a non-zero regularisation coefficient. On all case tested, we



(a) Image 1 (b) Image 2 (c) Ground truth disparity

(d) Left disparity field (e) Right disparity field (f) Consistency map

Figure 4.17: Left and right disparity fields after one iteration. On the left view, all PMBP steps were enabled. On the right view, only the view propagation from the left view were enabled, and the propagation and randomisation were disabled.

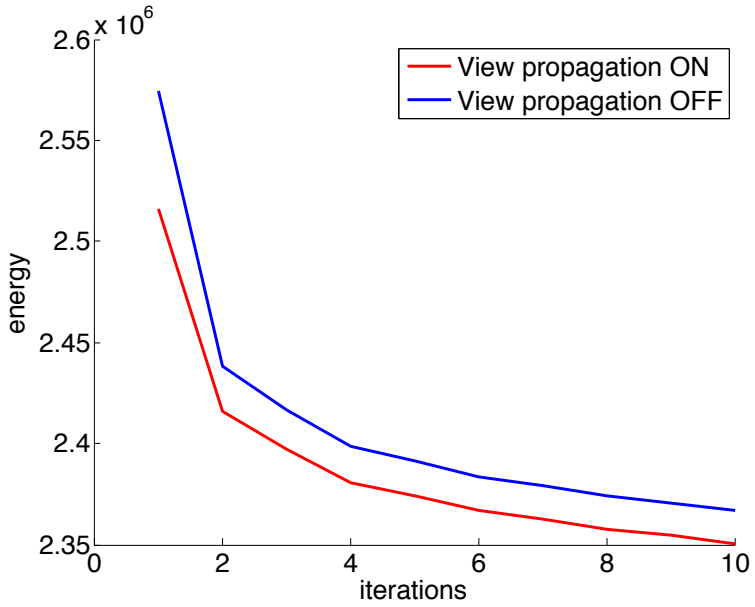


Figure 4.18: Comparison between PMBP with and without view propagation. Note that we omit the energy of the initial iteration as it is significantly higher, and equal for both curves.

observed that adding a certain amount of regularisation always produce higher quality results.

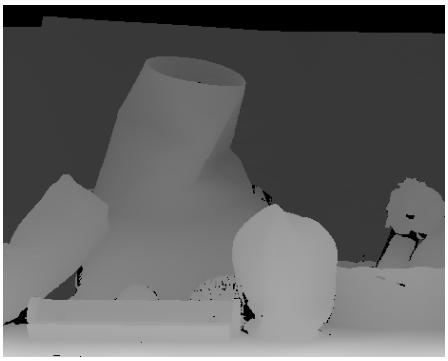
Our algorithm however has some limitations. First, it has a significant number of parameters which influence the results. Different datasets might have different optimal parameters, and therefore having a high number of parameters to tweak increases the difficulty to optimally use our algorithm. These include the regularisation level, the adaptive support weight, the different truncations terms, the patch size, the number of particles, the number of iterations and the weight of the gradient and colour values in the unary term. However, most of the parameters that we have used across our experiments, which have been determined empirically, were constant, and produced satisfactory results. In fact, we have used the same parameters for the optical flow application presented in the next chapters, which includes new datasets.

Second, it is currently not suitable for real-time usage, as a typical run for a 450×375 image takes 29 seconds for 4 iterations with 5 particles, for a small patch size of 5x5. While it is relatively efficient, there would be some extra approximations and tradeoff that would need to be made in order to reach real-time capabilities. Finally, the problems linked to low number of particles which we have describe in chapter 3 also apply here as PMBP is core to our stereo process, which means that in this case the result of the disparity estimation in textureless areas can be lacking quality. This can be seen in figure 4.19, where the scenes exhibit non-textured surfaces..

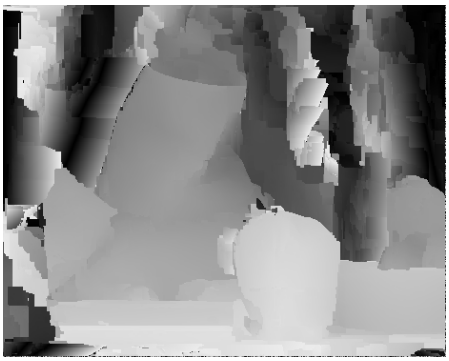
Future work includes different steps to address the limitations mentioned above. A training process could be undertaken in order to learn the optimal parameters and limit the amount of manual tweaking. Reducing the number of parameters is also an option, with for example the search for a simpler unary term, which by itself currently contains a high number of parameters. A third possibility would be to introduce parameters whose values are more meaningful and expressed in understandable units (for example a truncation term expressed in pixels), which could be easily chosen by the user. In the view of obtaining a real time algorithm, a coarse-to-fine scheme could be added. Typically, this is done by subsampling the images to create a multiscale pyramid. The algorithm is run on the top levels of the pyramid, which is efficient due to the size of the images at this scale. Several upsampling steps follow, where each time the solution found at lower scales are used to constraint the solution at the current scale. This has the effect to constrain the search space, which ultimately drastically improves the performance. Also, large displacements can be more easily estimated at lower scales. Another possibility would be to use extra information to improve the initialisation. For example, one could perform a SIFT matching operation [Lowe, 2004] in order to seed the initialisation in the neighbourhood of those feature points. Finally, our algorithm seems to be well suited to be run on videos, in the same fashion as [Bleyer et al., 2011]. An advantage of PMBP is that in this case the results could also benefit from a regularisation term added on the temporal dimension, in order to add coherence of the disparity maps generated over time.

(a) Midd1 dataset - Left Image

(b) Midd1 dataset - Right Image



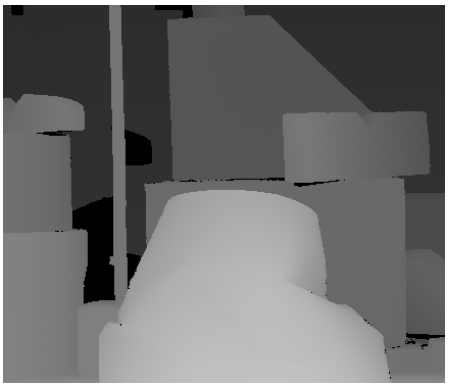
(c) Midd1 dataset - Ground Truth



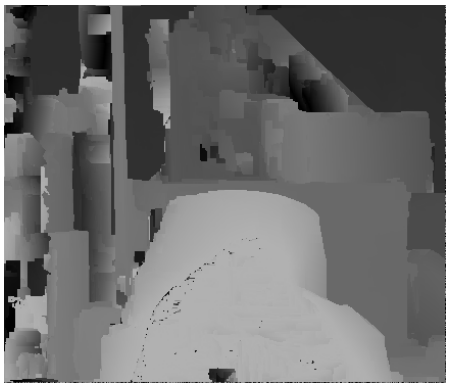
(d) Midd1 dataset - PMBP result

(e) Lampshade1 dataset - Left Image

(f) Lampshade1 dataset - Right Image



(g) Lampshade1 dataset - Ground Truth



(h) Lampshade1 dataset - PMBP result

Figure 4.19: Example of failure cases. Our algorithm, running with 1 particle, fails at producing a smooth result on the large white wall for the Midd1 dataset and on the yellow surface on the Lampshade1 dataset.

Chapter 5

Optical flow

5.1 Introduction

The optical flow computation problem is of great importance, as it is one of the core mechanisms of many machine vision methods. In particular it is commonly used for tracking [Decarlo and Metaxas, 2000] and video editing [Zitnick et al., 2005]. While this has been an active research field, there is almost no mention of PatchMatch-based methods for optical flow as mentioned in Chapter 2. However, PatchMatch, and in turn PMBP seem to be suitable optimisation algorithms to solve the optical flow problem, due to their efficiency, and the spatial and temporal natural coherency of pixels in a video. In this chapter we present our different optical flow models and describe how we take advantage of the PMBP framework when building them. First, in section 5.2 we analyse the advantages of using the original PatchMatch algorithm to perform optical flow computations with a 5D model including translation, rotation and anisotropic scaling in an efficient way which we apply to video editing. Note that this does not include any regularisation - it presents our motivation for using PatchMatch as an optical flow algorithm. Then, we show in section 5.3 that we can improve the quality of the flow results by switching to PMBP instead, which includes an explicit regularisation term in the energy. Finally, in section 5.4, we propose to further over-parameterise the search space and define an affine model (6D) to represent the motion between two images. This can be seen as a direct extension of our stereo model described in Chapter 4 and it aims at better capturing the disparities of slanted surfaces than the previously defined 5D model.

5.2 5D Optical Flow with PatchMatch

As mentioned in the previous chapters, PatchMatch is very efficient at optimising the unary cost function for correspondence field estimation tasks. In this section, we analyse the possibility or performing optical flow computations using a variation of PatchMatch adapted to matching consecutive images taken from a video. In addition, we design an easy-to-use application where the user can edit the first frame of the video, and we propagate the changes to the rest of the frames.

5.2.1 Methodology

Similarly to the 2D-PCE defined in section 3.1, we propose a unary cost function that calculates an error between two patches. However, while 2D-PCE only uses states representing a 2D offset to refer to

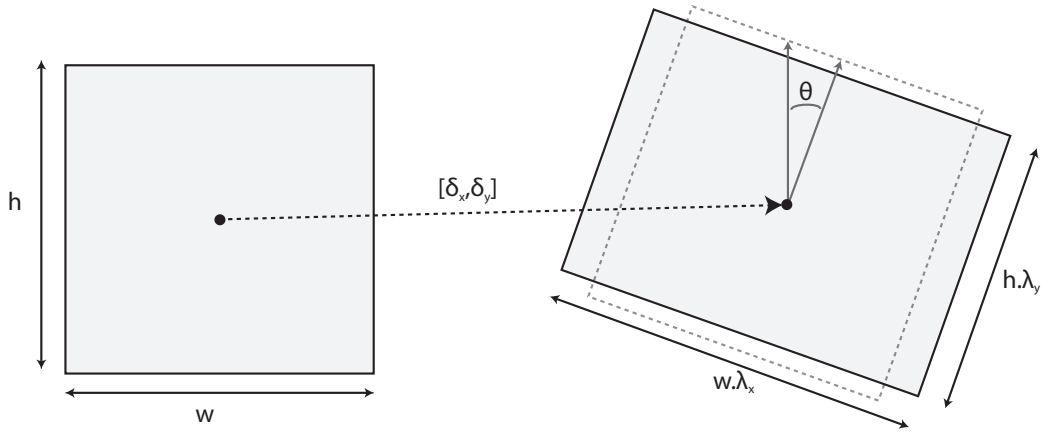


Figure 5.1: Patch warping function. The patch is first translated by $[\delta_x, \delta_y]$, then rotated by θ around its centre, and finally scaled by $[\lambda_x, \lambda_y]$. w and h are the dimensions of the original patch, which is usually square and verifies $w = h$.

the matching patch, we use a more complex model, that incorporates in-plane rotation and anisotropic scaling. Thus, we define for node s a state $\mathbf{u}_s = [\delta_x, \delta_y, \theta, \lambda_x, \lambda_y]$ where $[\delta_x, \delta_y]$ is a 2D offset, θ is the in-plane rotation angle, and $[\lambda_x, \lambda_y]$ is the scaling factor in the x and y direction respectively. We then define the unary cost function as:

$$\psi_s^{\text{wpf}}(\mathbf{u}_s) = \sum_{i=-h}^{h} \sum_{j=-h}^{h} w_{sij} \big\| I_1(x_s + i,\ y_s + j) - I_2([x_s + i, y_s + j] + f_{\mathbf{u}_s}([x_s + i+, y_s + j]) \big\|_\theta. \tag{5.1}$$

where $\mathbf{u}_s$ is the candidate state, h is the patch size, I_1 and I_2 the two images, w_{sij} the adaptive support weights defined in section 4.2.1, the norm used is the one also defined in section 4.2.1 and $f_{\mathbf{u}_s}$ is the *warp function* that applies the offset, rotation and scaling to a patch. To do so, when transforming the pixels of a patch, we consider the vectors joining these pixel with the centre of this patch. The warping function then operates by rotating and then scaling these vectors to compute the final positions of the pixels within the patch, before applying the final translation offset. This is illustrated in figure 5.1. Below we describe several important points concerning the algorithm.

Search Space

It is possible to set constraints on the search space as it improves the quality of the results, provided that the solution lies within the specified range. In our method, we set the following contraints:

- The displacement of the centre pixel $[\delta_x, \delta_y]$ is constrained to lie within the range $[-w, w] \times [-h, h]$ where w and h are the width and height of the target image respectively.
- The rotation angle θ is not contrained, and can take value in the range $[0, 2\pi]$.
- The scaling factors $[\lambda_x, \lambda_y]$ are contrained to the interval $[\frac{1}{\lambda_{\text{max}}}, \lambda_{\text{max}}]^2$ where λ_{max} is the maximum scaling factor and is usually set to 2.

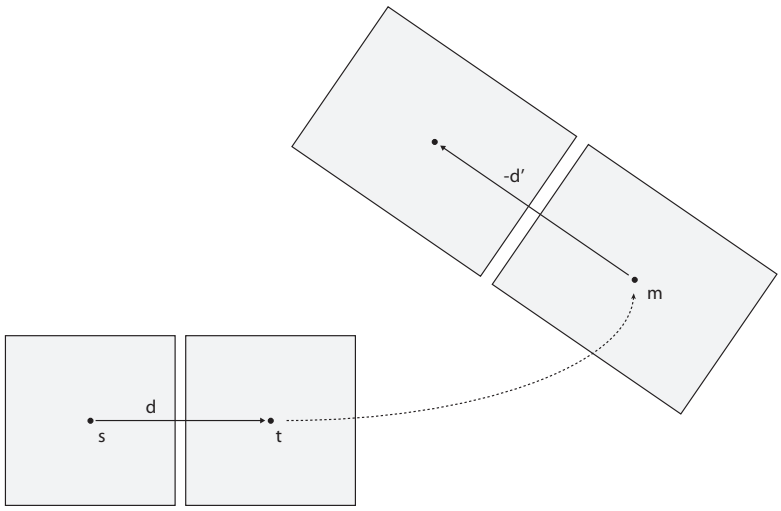


Figure 5.2: Illustration of the propagation in the 5D framework. d is the offset between two neighbouring patches s and t. If patch t matches with a patch m with a certain rotation and scale, this rotation and scale needs to be taken into account during the propagation step, and we see that a good candidate patch for s is the patch located at an offset $-d'$ of m, with the same rotation and scale as m.

Initialisation

Each parameter of each state is uniformly sampled within the allowed range defined above. For each random state that is generated, we ensure that it corresponds to a match that lies within the source image, and if not we reject it and sample a new state until we draw one that satisfies the conditions.

Propagation

When we use the states of a node t as candidate particles for a neighbouring node s, we use the offset between nodes s and t to recover the position of the potential candidate, however this offset needs to be transformed first. This is due to the fact that the rotation and scaling affect the offset of the new candidate particle, as can be seen in figure 5.2. Let us consider the state at node t, $u_t = [\delta_x, \delta_y, \theta, \lambda_x, \lambda_y]$. The neighbouring node s will be proposed the state u_s, which is an adjustment of the state $u_t = [\delta'_x, \delta'_y, \theta', \lambda'_x, \lambda'_y]$ such that:

$$[\delta'_x, \delta'_y]^\intercal = [\delta_x, \delta_y]^\intercal + [x_t - x_s, y_t - y_s]^\intercal - \begin{bmatrix} \lambda\cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \lambda\cos(\theta) \end{bmatrix} \begin{bmatrix} x_t - x_s \\ y_t - y_s \end{bmatrix} \tag{5.2}$$

$$\theta' = \theta \tag{5.3}$$

$$[\lambda'_x, \lambda'_y] = [\lambda_x, \lambda_y] \tag{5.4}$$

Randomisation

Given an existing particle u_s, we generate random samples around u_s. This is done by using a value r which exponentially decreases from 1 to 0, and then by generating new states $u'_s = [\delta'_x, \delta'_y, \theta', \lambda'_x, \lambda'_y]$ as

follows:

$$[\delta'_x, \delta'_y] = [\delta_x, \delta_y] + r[\hat{\delta_x}, \hat{\delta_y}] \tag{5.5}$$

$$\theta' = \theta + r\hat{\theta} \tag{5.6}$$

$$[\lambda'_x, \lambda'_y] = [\lambda'_x, \lambda'_y] + r[\hat{\lambda_x}, \hat{\lambda_y}] \tag{5.7}$$

where $\hat{\delta_x}$ and $\hat{\delta_y}$ are drawn from $U(-w, w)$ and $U(-h, h)$ respectively, $\hat{\theta}$ is drawn from $U(-\pi, \pi)$ and $\hat{\lambda_x}$ and $\hat{\lambda_y}$ are both drawn from $U(\frac{1}{\lambda_{\text{max}}}, \lambda_{\text{max}})$.

Sub-pixel Accuracy

While the original PatchMatch works with integer coordinates, we need to use sub-pixel accuracy for our matching, which is common practice for optical flow computations. Working with videos does indeed require an accurate matching across the images. Matching errors might be undetectable when looking at a still frame, however they are more obvious over a sequence of images. Because our matching is also used to perform tracking across several frames in a row, the errors due to a non-subpixel matching would accumulate over time and cause distortion and jitter. To avoid such artefacts, the position of matching pixels are naturally evaluated as floating point numbers. This provides a much finer grid than the one considering only integer coordinates. Furthermore, we maximise the precision of the other parameters in the search: scale and rotation, which we also represent as floating point values. It is worth pointing out that the Generalised PatchMatch algorithm [Barnes et al., 2010] discretises the values of rotation and scaling, while we use the full floating point range of available values. Due to this subpixel accuracy, the match of a given patch is not necessarily located at a grid position (i.e. centred on image pixels) any more. It can also be scaled and rotated by any arbitrary value. Consequently, we use a bilinear interpolation scheme to evaluate the colour of a pixel at non-integer coordinates.

Anisotropic scale

In a video, the scene is composed of 3D geometry that evolves over time. Our goal, by choosing this particular model, is to approximate the motion of the 3D patches of the scene by 2D patches. To do so, and to provide a wider range of deformation, we separate our scaling parameter into two distinct components: scale along the X direction and scale along the Y direction. This allows a better approximation of distortions as patches can be stretched or squeezed in one direction only. This type of deformation of the 2D image can often be seen due to the perspective foreshortening effects.

Barnes et al. [Barnes et al., 2009] use a jump flood algorithm for their GPU implementation. To speed up the matching process, we use a similar idea to Barnes et al. [Barnes et al., 2010], which is suited to multi-core CPU usage. Given n available cores, we split the image into two (shifted) sets of n horizontal bands equally distributed across the image. At odd iterations, each core operates independently on a horizontal band from the first set and at even iterations, each core operates independently on a horizontal band from the second set. While there is no propagation of information across bands during a single iteration, the following one will propagate it, as the seam is offset from the previous iteration.

5.2.2 Tracking

Our editing paradigm is rather simple: a user provides an edited first frame of the video, whose edits are then propagated to the rest of the images. To this end, we need to follow the patches that have been modified over the video sequence. Our input video contains k frames, where we need to propagate the modifications made on frame 0. There are a number of possible ways to achieve this. Here we present three tracking schemes, each one having its own strengths and weaknesses. Let f_n denote the n-th frame of the video. Let $T : p, n \longmapsto T(p, n)$ be the 2D transformation representing the position, orientation and scale of a patch p in f_n and let $F : p, n, m \longmapsto F(p, n, m)$ be our tracking function, representing the 2D transformation of patch p between f_n and f_m.

Tracking scheme 1: Initial and current image

The first method that we define to track patches is to match the initial frame of the video with all other frames, such that for any patch p, its position in f_n is given by:

$$T(p, n) \quad = \quad F(p, 0, n) \tag{5.8}$$

As mentioned above, we have control over the initialisation of our matching algorithm, and this first step is very important as it provides the input to the first propagation step in the process. While random initialisation works well when there is no prior on the pair of frames to match, we know in our case that there is a very strong correlation between the frames of a video. This means that initialising the NNF between f_0 and f_n with the results of the matching between f_0 and f_{n-1} already provides a good approximation. This adds robustness and leads to a quicker convergence than using a random initialisation.

This method works best on scenes where the objects are rigid and do not undergo significant deformation nor motion. This is expected, as the allowed 2D transformation of a patch cannot express the possible changes a patch can undergo from frame 0 to frame n that corresponds to a late time point in the sequence.

Figure 5.3: Comparison of using different tracking schemes. On the left, blur and wobbling is introduced over time by matching consecutive frames due to drift. On the right, we show that using reference frames results in fewer artefacts.

Tracking scheme 2: Consecutive images

An alternative way to perform tracking is by matching each pair of consecutive frames. We start with frames (f_0, f_1), and then match each pair (f_n, f_{n+1}), concatenating the 2D transformation of each patch that we are interested in. In other words we recursively track p, as follows:

$$T(p, n) \quad = \quad T(p, n-1)F(p, n-1, n) \tag{5.9}$$

Under the assumption that the motion between consecutive frames is small, we choose to initialise the fields of matches with identity transformations, as they are likely to be good approximations of the actual NNF.

This tracking scheme can cope with cases where the previous method fails; if the scene is changing too quickly, it is better to track patches by only matching consecutive images, so that the change in appearance or pose of each patch is minimised. However, drift is more likely to occur because concatenating many transformations results in a faster accumulation of the error.

Tracking scheme 3: Reference and current image

The third method is a good trade-off between the two previous ones and is the one we use in our applications. We use a reference frame f_r to match against the other frames, and update f_r regularly as we advance in the video. Let k be the number of frames that we track using f_r before updating r. We have:

$$T(p, n) \quad = \quad T(p, r)F(p, r, n) \tag{5.10}$$

Assuming that the current frame is f_n, we update r such that $r = n$ when $n - r >= k$.

It generates less drift caused by the accumulation of small errors, and is able to track patches that change in pose or appearance better than the first method. A comparison of the blur artefacts generated when using the second and third tracking scheme is illustrated in figure 5.3. We can see that using a reference frame is preferred over matching consecutive pairs of frames, as the quality of the output frames is higher.

Edit Propagation from Frame to Frame

The previous step gives us a set of 2D transformations, mapping the location of the originally edited patches to their updated locations in all the future frames. The last step of our framework is to apply these transformations to the original patches to propagate the modifications on each frame.

For this reason, we adopt a simple blending scheme. We apply on each originally edited patch its 2D transformation at the frame that is being processed and paste it onto the output image. As many patches will be superimposed, each of them casts a "vote" regarding which colour should be synthesised at each pixel. After transforming all the edited patches, we average the colours according to the number of votes, and output the final, modified image.

5.2.3 Applications

Our optical flow algorithm, combined with a tracking method can be used to perform a number of different editing tasks on videos, which we describe below.

Painting

Painting the surface of an object can be performed on the first frame of the video with any image editing software, and then plugged into our system to generate the final output. This is illustrated in figure 5.4.

Our algorithm requires only one input: the modified first frame of the video. We then compute a mask indicating which pixels have been modified automatically by differencing the original and the edited image.

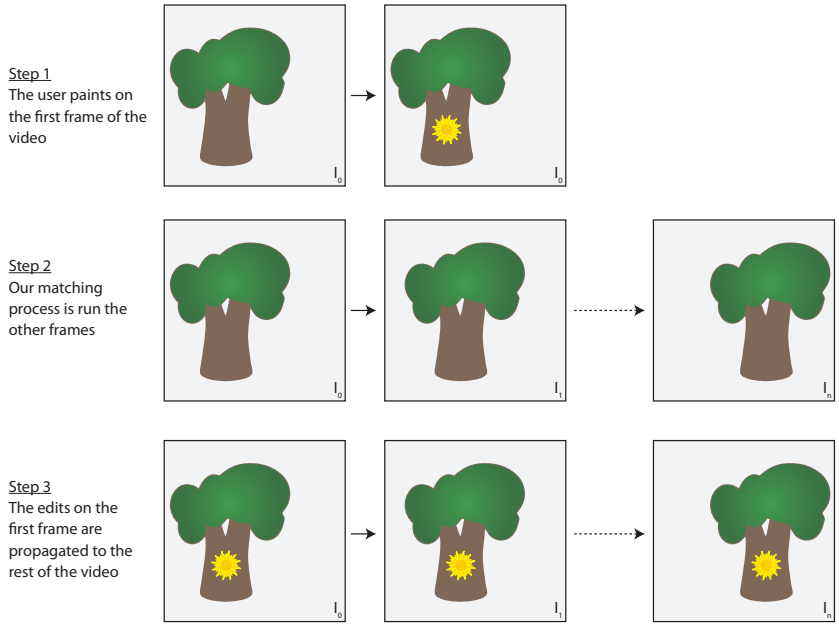
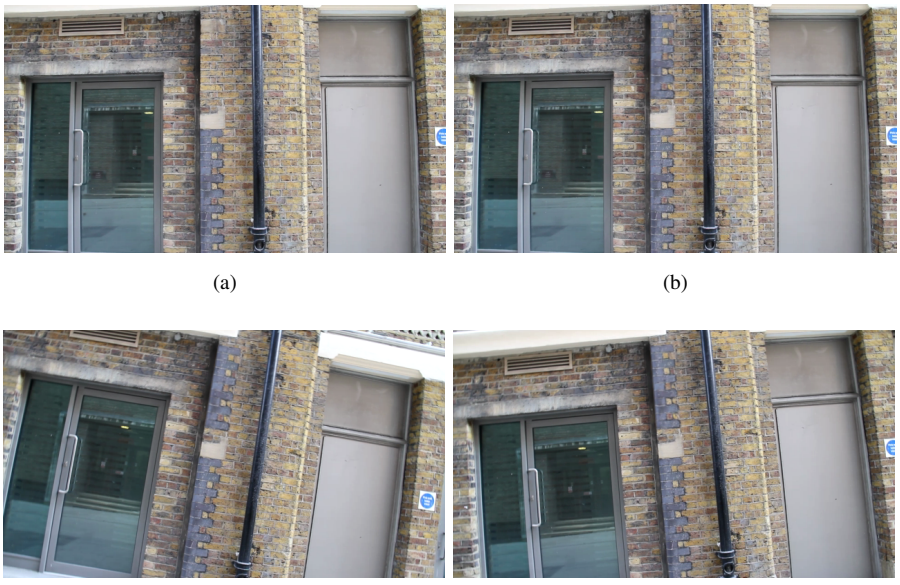


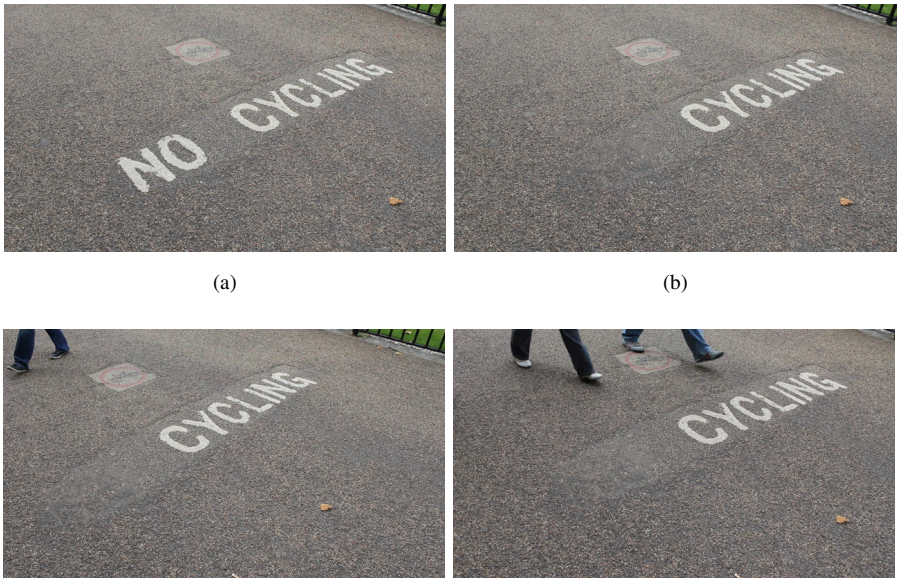
Figure 5.4: Our typical workflow for painting over a video explained in three steps. This can also be applied to inpainting, where instead of painting over the first frame of the video, we use an inpainting algorithm to make the changes to this first frame.



(a) (b)

(c) (d)

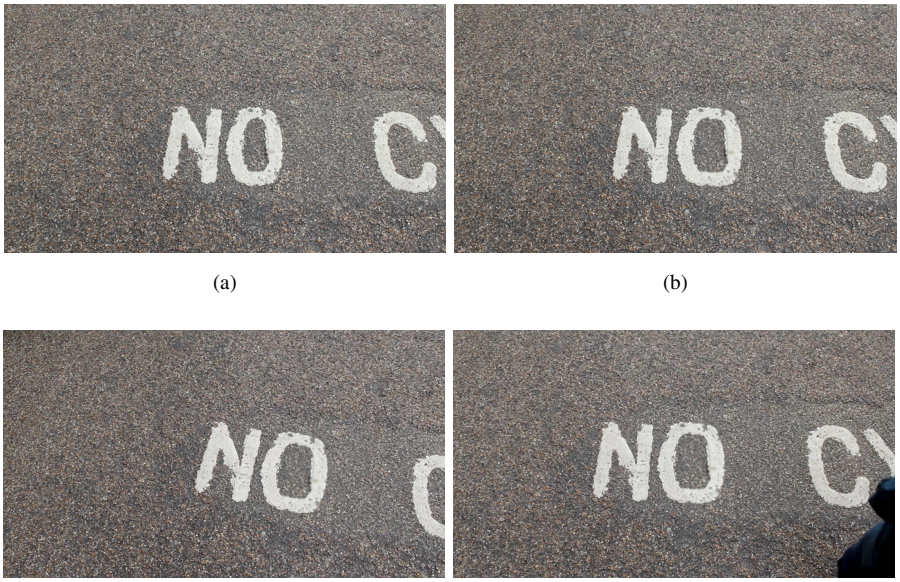
Figure 5.5: Painting example on a planar surface. (a) First frame of the original video; (b) Blue bricks have been added on a section of the wall; (c) & (d) Images of the video where the edit has been propagated with our algorithm.



(a) (b)

(c) (d)

Figure 5.6: Inpainting example. (a) First frame of the original video; (b) The word ”NO” has been removed; (c) & (d) Images of the video where the edit has been propagated with our algorithm.



(a) (b)

(c) (d)

Figure 5.7: Deformation example. (a) First frame of the original video; (b) The letter ’O’ has been enlarged; (c) & (d) Images of the video where the edit has been propagated with our algorithm.

In section 5.2.4 we demonstrate how this can be used to paint over both planar surfaces (e.g. wall), and non-planar convex surfaces (e.g. part of a face, rock).

Inpainting

Inpainting is the process of synthesising missing parts of an image in a realistic way. It can be typically used to correct defects, or remove unwanted objects. We use our matching algorithm to perform inpainting, following the technique of Barnes et al. [Barnes et al., 2009].

We perform three different types of inpainting tasks.

Object removal The first inpainting application is object removal, ie. removing unwanted elements from the scene. The user creates a mask to select the area that needs to be inpainted. Our algorithm calculates which patches should be pasted in order to fill this newly created hole, and outputs an initial inpainted image, similarly to the original PatchMatch algorithm. We can then propagate this synthesised area to the other frames of the video.

Object deformation We provide tools to the user to select an area of his choice and displace it. This can be used to perform rigid object deformation, such as shortening a building, enlarging a window or a door, etc.

Object displacement Finally we propose object displacement where the user can select the whole object and move it somewhere else in the image. We then inpaint the hole left by the object in its original location, and also inpaint a band around the object at its new location in order to blend it in correctly with its new surroundings. All the inpainted pixels are then propagated to the rest of the video.

5.2.4 Experiments

In this section we present the results of several editing tasks, described above, that we carried out on real footage.

Planar objects

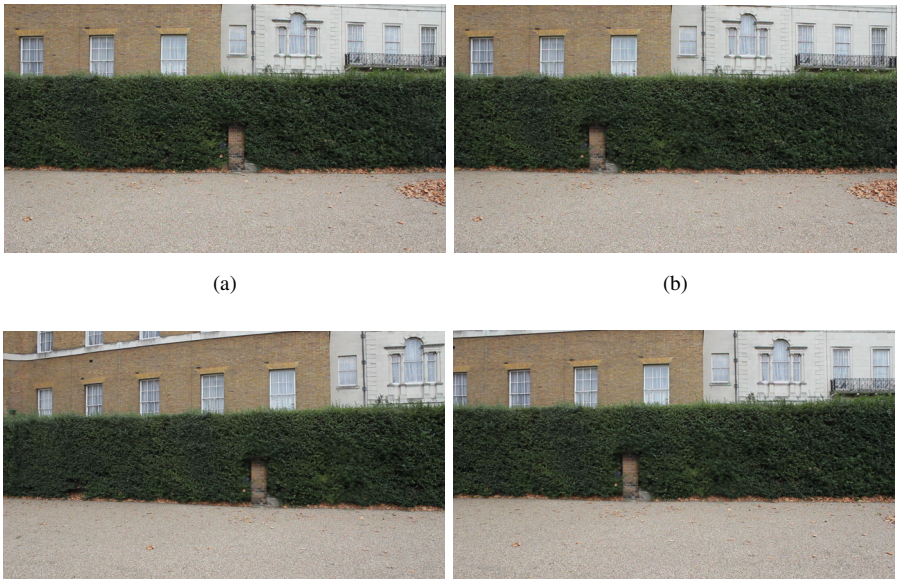
We first demonstrate that our approach can handle planar objects.

The wall sequence We present the result of a painting task on the footage of a wall. Here we copy a part of the blue bricks onto the red bricks located above. The camera's motion incorporates translation and rotation. Snapshots of the resulting video can be seen in figure 5.5. As we can see, there are no apparent artefacts and the result looks realistic.

The "no cycling" sequence We performed two different edits on a video showing a road with "no cycling" painted on the ground.

First, we removed the word "no" using our inpainting algorithm, as shown in figure 5.6. As it can be seen, our approach handles the camera rotation well. This kind of modification allows the user to remove unwanted elements from the scene, despite arbitrary camera motion.

Then, we applied a deformation and widened the letter 'o', as seen in figure 5.7. The background is textured enough to allow our algorithm to propagate this edit to the rest of the sequence without any artefacts.



(a) (b)

(c) (d)

Figure 5.8: Object displacement example. (a) First frame of the original video; (b) The brick door has been dragged further left on the bush; (c) & (d) Images of the video where the edit has been propagated with our algorithm.

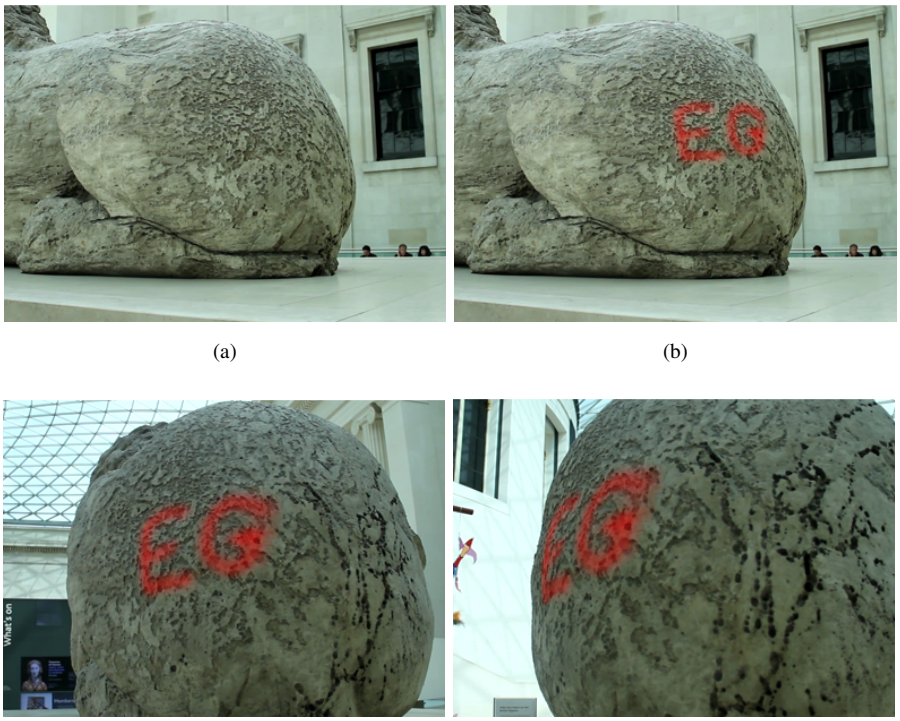
The bush sequence In this sequence we illustrate the task of object displacement. The video shows a bush with a brick door in the middle. We used our system to move the door to the left of the scene. Snapshots of the output video are shown in figure 5.8. This result required two steps from our algorithm: first inpainting the hole left by moving the door, and also resynthesising a band around the new shifted door in order to blend it realistically with the background. This type of modification gives a compelling effect when applied to a video. This has the potential to provide a very straightforward ”drag and drop” video editing tool to the users.

Non-planar objects

We further demonstrate that our approach can handle non-planar as well as deformable objects.

The statue sequence In this sequence, we filmed the statue of a lion while moving the camera closer and on a circular motion around it. We painted an “EG” graffiti onto its side. There are several interesting observations to be made about this sequence. First, the shape of the statue is not planar, and displays some irregularities on its surface. Furthermore, there is a significant perspective distortion between the first and the last frame, and the camera is moving closer to the statue. Finally, the illumination changes: the lion appears darker towards the end, as the camera is facing a bright background which increases the contrast of the scene.

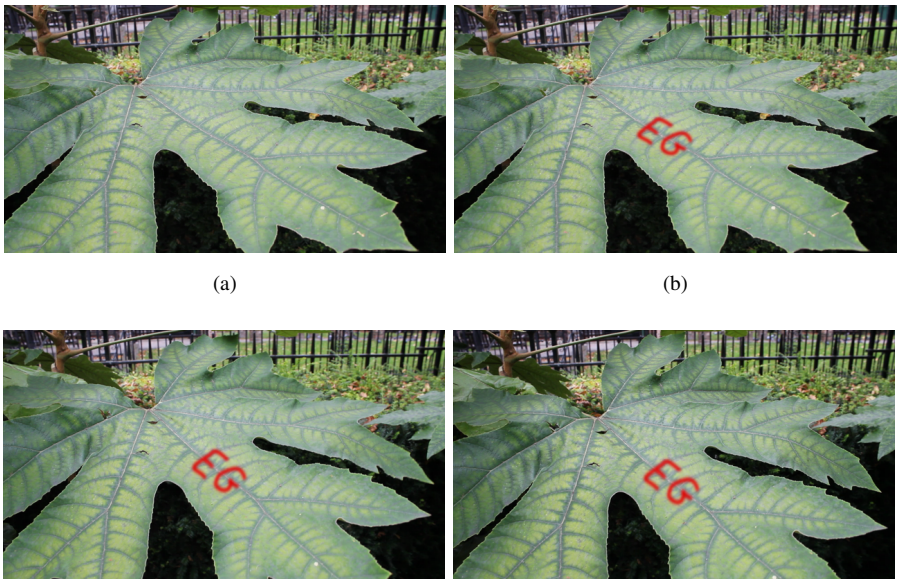
Figure 5.9 shows the result of this footage. Our algorithm handles all these difficult points well, and the result appears realistic. There is no drift, and the patches undergo the correct perspective distortion. As the change in illumination is smooth, we are able to keep track of the patches even though their appearance changes significantly between the first and the last frame.



(a) (b)

(c) (d)

Figure 5.9: Example of painting over a non-planar surface with a camera motion yielding perspective distortion. (a) First frame of the original video; (b) User-modified frame: the letters ”EG” have been painted over the statue; (c) & (d) Images of the video where the edit has been propagated with our algorithm.



(a) (b)

(c) (d)

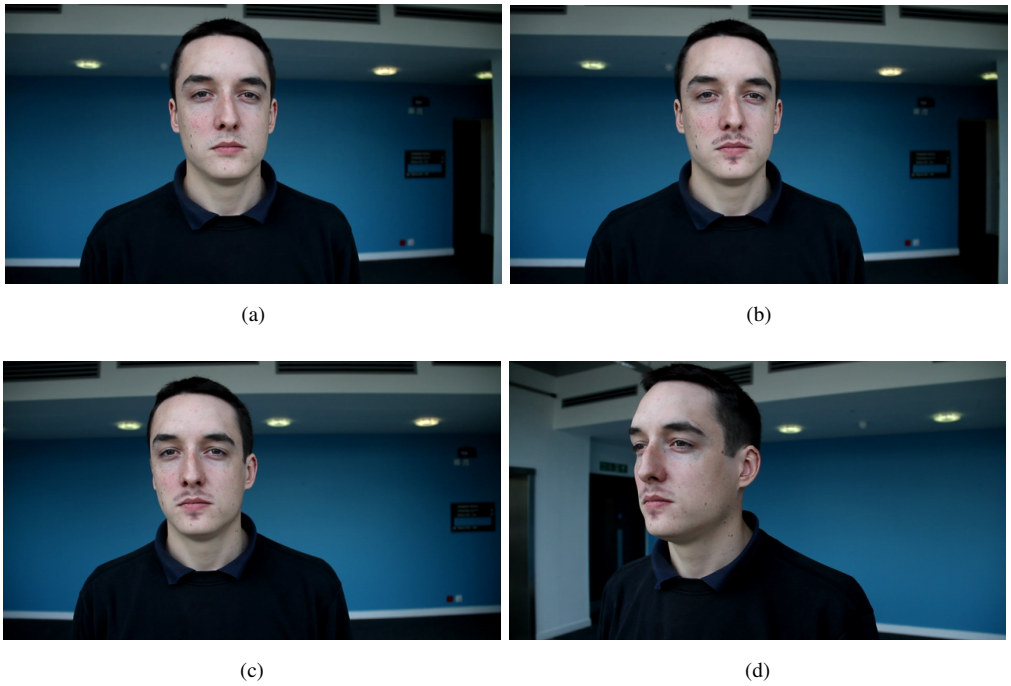
Figure 5.10: Painting example on a deformable object. (a) First frame of the original video; (b) The letters ”EG” have been painted on the surface of the leaf; (c) & (d) Images of the video where the edit has been propagated with our algorithm.

The leaf sequence To illustrate the algorithm's ability to handle dynamic scenes, we used a sequence of a large tree leaf blowing in the wind. The wind deforms the leaf, as it is a flexible object. We paint the letters “EG” on it. The results can be seen in figure 5.10. This is an interesting sequence as it is an example of a dynamic and deformable object. As the surface is stretched when the wind is strong, the modifications are stretched accordingly.

The face sequence We filmed the face of a person while moving around him, and then painted a moustache to the footage. Results can be seen in figure 5.11. This is another example of a non-planar surface, and the output appears realistic, with no apparent artefacts.

Performance

An unoptimised implementation of our algorithm requires about 5 seconds per frame on a 640x360 video on 2 cores of an Intel Xeon X5670 at 2.93Ghz, with 10 iterations of the matching process, using 7x7 patches. We believe that the code can be further optimised to reach higher speed. Indeed, [Barnes et al., 2009] demonstrate that a GPU implementation can match an entire 0.2 megapixel frame in about 0.2s (5 iterations, on an old NVIDIA 8800). Although our algorithm is more computationally expensive as we use bilinear interpolation to access image pixels, it should stay within the same order of magnitude. This means that our algorithm could potentially run significantly faster than our reported timings, which is already faster than its competitors.



(a) (b)

(c) (d)

Figure 5.11: Painting example on a non-planar surface. (a) First frame of the original video; (b) A moustache has been painted on the person; (c) & (d) Images of the video where the edit has been propagated with our algorithm.

Limitations

The main limitation of this method is that it struggle propagating the edits accurately to textureless areas. This is linked to the fact that this model does not use any regularisation term in the energy that is being optimised, and thus suffers from the weaknesses that we have established in Chapter 3. This can be seen in the experiments that we run in section 5.3.2, where we compare this method against a similar technique using PMBP as the optimiser to be able to include a regularisation term in the energy, and use ground truth optical flow displacement maps to quantitatively evaluate both methods. We also illustrate how this method compares with the other techniques presented in this chapter on several optical flow benchmarks in section 5.5.

5.2.5 Discussion

In this section we proposed using PatchMatch to perform optical flow computation, as it is an efficient way of establishing correspondence between two images. Furthermore, this framework can benefit from the fact that we are matching consecutive images in a video, as the initialisation of the solution is straightforward and provides good quality results. Furthermore, we observe a certain amount of temporal stability due to this process of initialising the solution for a frame with the solution at the previous frame. While it seems good enough for our application, it would be beneficial to include additional constraints in the optimisation to explicitly enforce temporal stability. The purpose of the previous section was to motivate the use of PatchMatch-related methods for dense optical flow, which has not been done in the

literature. We show that this method can be used to perform video editing tasks easily. We can keep track of patches located on a 3D surface without the need to reconstruct the 3D scene; we assume instead that a change of pose in 3D can be represented as a set of local 2D transformations. However it is unable to propagate the edits to textureless areas, and using an optimiser that can include an explicit smoothness term would be beneficial.

[Rav-Acha et al., 2008] obtain high quality results and perform similar tasks to our algorithm. In addition, they can handle occlusions in videos, which is an advantage compared to our technique. However, their algorithm requires a segmentation of the edited object throughout the sequence, a difficult problem that is also computationally expensive, and is not required in our algorithm.

The nature of the matching algorithm that we use is well-suited to track patches between frames of a video. as the propagation step of our approach gives a certain amount of coherence to the NNF, which is needed for tracking. However, the lack of explicit regularisation term in the optimisation is a drawback of this method, which could cause problems due to the reasons explain in Chapter 3. We address these by using PMBP as optimiser, instead of PatchMatch. This is explained in detail in the next section.

5.3 5D Flow using PMBP

In order to improve the method presented in the previous section, we integrate it into our PMBP framework and include a smoothness term to encourage the flow to be piecewise constant. In this section we describe this new pairwise term, and show how it improves the quality of the generated flow results.

5.3.1 Methodology

We introduce a pairwise term that encourages the states of neighbouring nodes to be similar. In our approach, the states contain the flow of the middle pixel of a patch, and in addition a rotation and scaling term. Therefore our pairwise term needs to reflect the similarity both in flow, and also in the patch transformation. To do so, we consider the displacement of the middle pixel, and in addition the displacement of two corners of the patch, as seen in figure 5.12, as two identical states have the same effect on the two vectors linking the middle pixel with these two corners. The pairwise term is mathematically defined as follows:

$$\psi_{st}(\mathbf{u}_s, \mathbf{u}_t) = \min\big(\kappa, |f_{\mathbf{u}_s}(s) - f_{\mathbf{u}_t}(s)| + |f_{\mathbf{u}_s}(s^{\text{tl}}) - f_{\mathbf{u}_t}(s^{\text{tl}})| + |f_{\mathbf{u}_s}(s^{\text{tr}}) - f_{\mathbf{u}_t}(s^{\text{tr}})|\big) \tag{5.11}$$

where $\mathbf{u}_s$ and $\mathbf{u}_t$ are two neighbouring states, s^{tl} is the top left corner of the patch centred around s, s^{tr} is the top right corner of the same patch, and κ is a truncation term. The model that we use has 5 degrees of freedom, so using the displacement of 3 non-colinear points is enough to fully capture the transformation difference.

5.3.2 Experiments

Our model has a number of parameters that can affect the results in different ways. We designed several experiments that show the effect of varying two of these parameters: the patch size and the smoothness value. Note that all our experiments in this section have been run with 5 particles for 4 iterations.

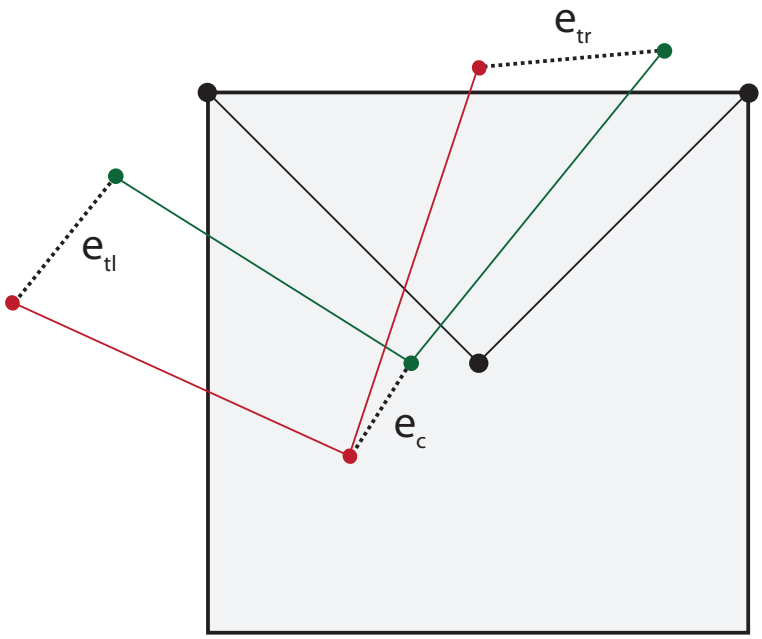
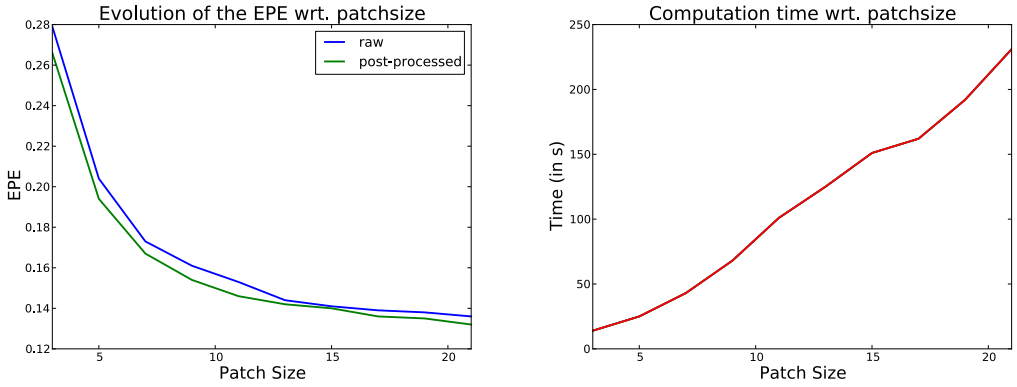


Figure 5.12: 3 point pairwise term. The original patch is shown in black. The red points show the original points transformed with the first state u_t, while the green points show the original points transformed with the second state u_s. The difference in point positions represent the three errors e_c, e_{tl} and e_{tr} which are added and then truncated.



(a) (b)

Figure 5.13: (a) Evolution of the End Point Error (EPE) with respect to the patch size; (b) Computation time with respect to the patch size

First, we keep the smoothness term constant and show the influence of the patch size on one case of the flow dataset of the Middlebury database [Baker et al., 2011], namely 'RubberWhale'. We vary the patch size from 3x3 pixels to 21x21, and the resulting EPE for both raw and post-processed outputs can be seen in figure 5.13. We observe that there is a direct relationship between the patch size and the EPE, which appears to decrease with larger patch size. This effect is reflected for both raw and post-

processed outputs. Naturally, increasing the patch size leads to higher computational costs, an effect that is illustrated in the same figure.

Next, we investigate the effect of the regularisation term on one particular case by varying the smoothness level and calculating the result EPE for each value. This is an experiment that we have also previously conducted in Chapter 4 to demonstrate the effect of adding a regularisation term to the optimisation for the stereo matching application. Here all the other parameters are fixed and are chosen empirically. The resulting errors are shown in figure 5.14. We can see that similarly to the results we had in Chapter 4, adding a certain level of regularisation is beneficial to the quality of the output. On this case, a value of $\lambda = 0.01$ seems to be optimal. From $\lambda = 0$ to $\lambda = 0.01$ the EPE seems to decrease almost monotonically, while for $\lambda = 0.01$ to $\lambda = 0.03$ the error increases. This is consistent with the stereo results, where we observed similarly looking curves.

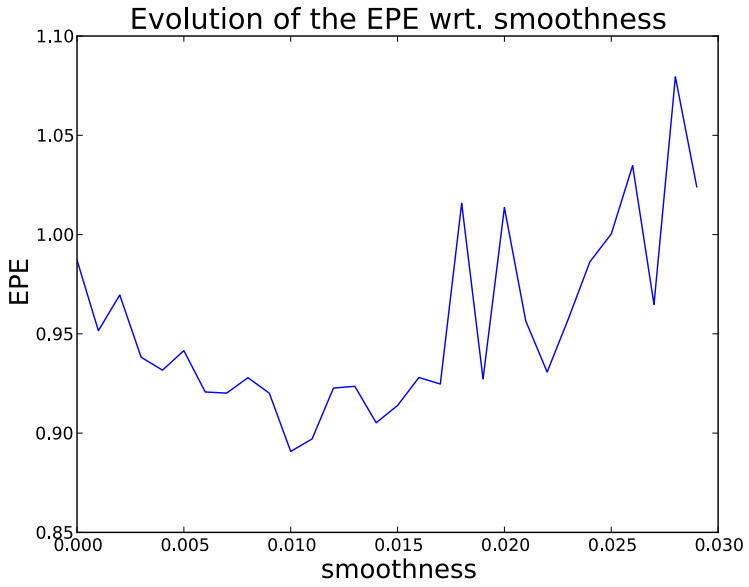


Figure 5.14: Evolution of the End Point Error (EPE) with respect to the smoothness term.

We carry out an experiment to validate the benefits of our regularisation term compared to the previous method. For this we use the Middlebury training Dataset [Baker et al., 2011] as well as the UCL dataset [Mac Aodha et al., 2012]. We run the algorithms both with and without regularisation term which is set to $\lambda = 0.01$ for PMBP. All the other parameters are set as in section 4.3.2, that is, $\{\omega, \alpha, \tau_{col}, \tau_{grad}\} = \{10, 0.9, 10, 2\}$, but with a patch size of 21×21. The results are displayed in table 5.1. Note that we run all the cases with the same parameters to establish a fair comparison. We observe that for most of the cases (26 out of 32), the method with regularisation term using PMBP performs better that the one without, presented in the previous section.

We run PMBP on frames from a video containing textureless areas. Results can be seen in figure 5.17. We can see that on half of the datasets, the solution does not look consistent with the expected motion on these flat areas, while on the other half the solution seems reasonable.

Finally, for completeness we show in figure 5.15 some results of the video painting application

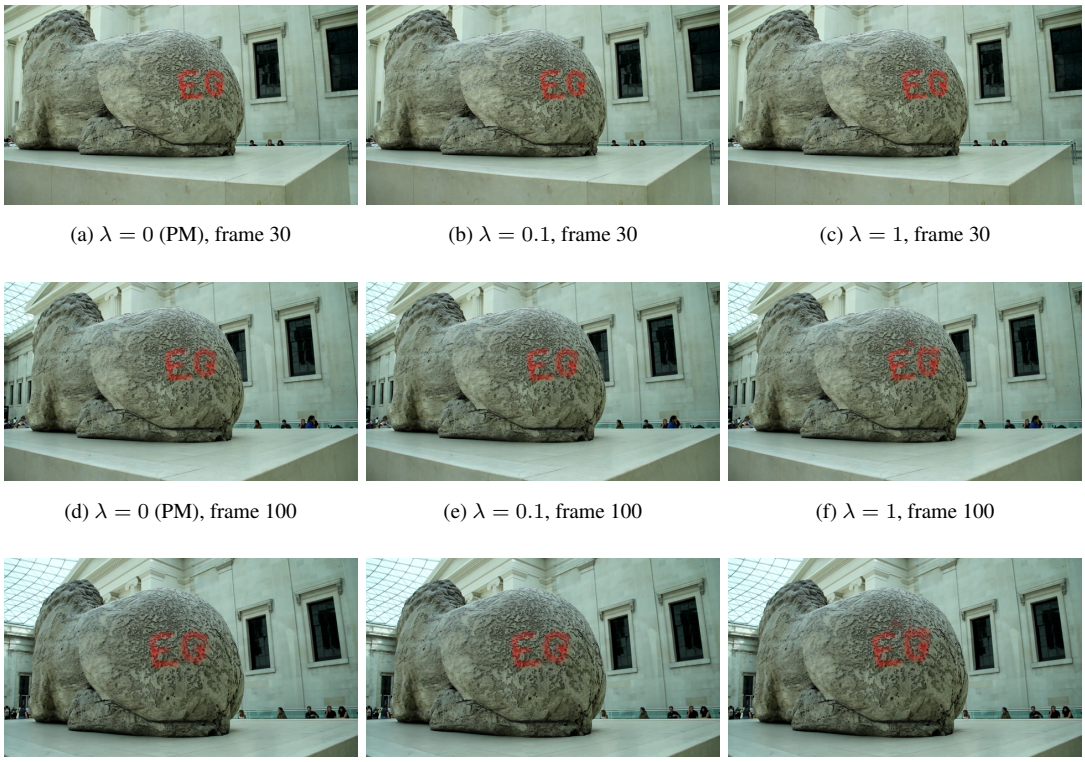
	5D PM	5D PMBP
UCL Lg. Disp.		
Crates1	2.96	**2.844**
Crates2	**2.32**	2.351
Mayan1	**0.454**	0.467
Robot	1.659	**1.377**
Crates1Htxtr2	0.5	**0.465**
Crates2Htxtr1	1.996	**1.033**
Brickbox1t1	0.33	**0.313**
Brickbox2t2	0.559	**0.548**
GrassSky0	**0.53**	0.542
GrassSky9	0.457	**0.392**
blow19Txtr2	0.23	**0.229**
drop9Txtr2	1.192	**0.982**
street1Txtr1	7.247	**6.883**
UCL Sm. Disp.		
Mayan2	0.324	**0.304**
YosemiteSun	0.7	**0.396**
GroveSun	0.308	**0.306**
Sponza1	1.92	**1.852**
Sponza2	**2.32**	2.4
TxtRMovement	**0.24**	0.484
TxtLMovement	**0.439**	0.471
blow1Txtr1	0.048	**0.047**
drop1Txtr1	0.063	**0.059**
roll1Txtr1	**0.002**	**0.002**
roll9Txtr2	0.016	**0.015**

(a)

	5D PM	5D PMBP
Middlebury		
Venus	0.336	**0.32**
Urban3	1.07	**0.777**
Urban2	0.332	**0.324**
RubberWhale	0.132	**0.13**
Hydrangea	0.206	**0.204**
Grove3	0.563	**0.558**
Grove2	0.218	**0.21**
Dimetrodon	0.19	**0.186**

(b)

Table 5.1: Comparison of EPE of the two 5D methods without (5D PM) and with (5D PMBP) smoothness term, on (a) the UCL data set and (b) the Middlebury dataset. Bold values indicate the best performance.



(a) $\lambda = 0$ (PM), frame 30 (b) $\lambda = 0.1$, frame 30 (c) $\lambda = 1$, frame 30

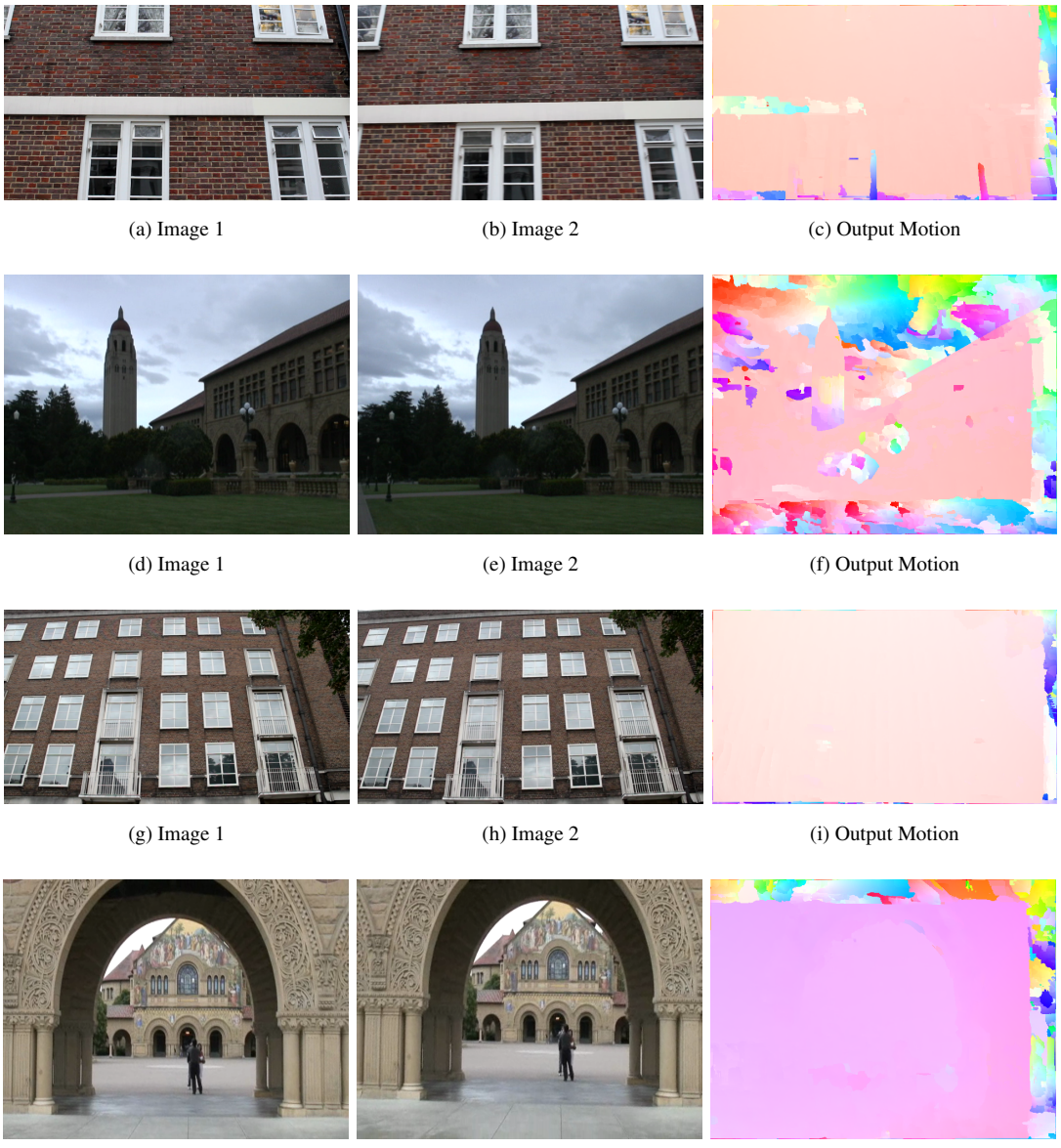
(d) $\lambda = 0$ (PM), frame 100 (e) $\lambda = 0.1$, frame 100 (f) $\lambda = 1$, frame 100

(g) $\lambda = 0$ (PM), frame 150 (h) $\lambda = 0.1$, frame 150 (i) $\lambda = 1$, frame 150

Figure 5.15: Comparison of the video editing application using PatchMatch ($\lambda = 0$) and PMBP. We can see that PatchMatch and PMBP with $\lambda = 0.1$ produce a similar result. Some artefacts can be seen on the result of PMBP with $\lambda = 1$ at frame 100 and 150.

(a) $\lambda = 0$ (PM) (b) $\lambda = 0.1$ (c) $\lambda = 1$

Figure 5.16: Comparison of the motion field using PatchMatch ($\lambda = 0$) and PMBP (with both $\lambda = 0.1$ and $\lambda = 1$).



(a) Image 1 (b) Image 2 (c) Output Motion

(d) Image 1 (e) Image 2 (f) Output Motion

(g) Image 1 (h) Image 2 (i) Output Motion

(j) Image 1 (k) Image 2 (l) Output Motion

Figure 5.17: Example of motion output by PMBP on real scenes that exhibit textureless areas. The textureless areas on these datasets are: (Row 1) the white band across the facade; (Row 2) the sky and grass; (Row 3) the inside of each window; (Row 4) parts of the sky, floor and columns. We can see on the datasets depicted in row 1, 2 the algorithm does not find a solution consistent with the rest of the image at these areas, while the solutions of row 3 and 4 look reasonable.

using PMBP, instead of the method devised in section 5.2. As the areas chosen for painting were fairly textured, the PatchMatch method performs visually as well as the PMBP method, while the latter starts displaying artefacts when the regularisation coefficient is set too high. This is due to the fact that any error is amplified over time due to the temporal propagation of the first edit. We also show the motion field for some of the matched frames in figure 5.16 where we can see that the motion output by PMBP seems less noisy than that of PatchMatch.

5.3.3 Discussion

In this section we have demonstrated a clear improvement on the quality of the flow field when using PMBP, as it produces lower End Point Error. This behaviour was expected, as we introduced a pairwise term that penalises two different neighbouring states by analysing the produced displacements for three points of each patch. However, this model assumes a piecewise constancy of the flow and of the patch transformation in general, which is not reflecting the real underlying 3D geometry. This can be easily understood, especially in the case of large displacements which often induce a radical change of appearance due to the perspective distortion involved. Our model is unable to cope with these cases and capture the true deformation, which cannot fully be expressed as a composition of translation, rotation and scaling only. We attempt at addressing this problem by over-parameterising further the flow field in the next section where we use an affine model.

5.4 Optical Flow with an Affine Model

One of the main drawbacks of the method described in the previous section is that is uses the piecewise constancy assumption, which is not accurately reflecting the geometry of the scene, as for slanted (non fronto-parallel) surfaces, a smooth variation of the flow is expected, rather than a constant flow that the previous pairwise term encourages. For the stereo application, this problem has been tackled in the literature by keeping the same 2D model and using higher order energy terms [Woodford et al., 2009], or by increasing the dimensionality of the search space to represent more accurately the geometry of the scene, such as the method that we used in Chapter 4. For optical flow, a similar method can be applied, and we show in this section how we can get a direct extension of the method used in Chapter 4 applied to optical flow. We achieve this by over-parameterising the search space into 6 dimensions we can get a direct extension of the method of Chapter 4 applied to flow, and using an affine model. Affine models have been used in the past for optical flow [Nir et al., 2008], and we show how this model can be adapted to the PMBP framework by using 4D disparity planes to allow for convenient propagation and randomisation of the states.

5.4.1 Transformations

We first introduce the notations and transformations that we use throughout the rest of this chapter.

We denote by $\mathring{\mathbf{s}} := [\mathbf{s}^\intercal 1]^\intercal$ the homogeneous representation of a 2D point $\mathbf{s}$, and define $\pi(\mathring{\mathbf{s}}) = [x_s/z_s, y_s/z_s]^\intercal$ and $I(\mathring{\mathbf{s}}) = I(\pi(\mathring{\mathbf{s}}))$. We introduce three types of geometric transformations that are represented with matrix notations.

Translation The translation is the simplest type of transformation which we have already used in the 2D-PCE application defined in Chapter 3. It is defined as follows:

$$T = \begin{pmatrix} 1 & 0 & \delta x \\ 0 & 1 & \delta y \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.12}$$

and has two degrees of freedom. Translations are rigid transformations that do not change the shape of a patch.

Affine The affine transformation is not a rigid transformation as it includes scaling and shearing in addition to translation and rotation. Its main property is that it preserves parallelism, which means that parallel lines remain parallels once transformed. It is defined as:

$$T = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.13}$$

and has 6 degrees of freedom. Note that the coefficients c and f are the translation coefficients, while the rotation, scaling and shearing are all included in the four other coefficients.

Homography The homography transformation has 8 degrees of freedom, and is defined as:

$$H = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix}. \tag{5.14}$$

The main property of homographies is that they can accurately represent the transformation of a plane in image space between two images, which is explained more in detail in Chapter 6.

Notations For a transformation M, we define the operator $*$ such that $M * \mathbf{s} := \pi(M\mathring{\mathbf{s}})$. In our framework, a pixel (or node) s is associated with states with notations u_s, as we have seen in the previous chapter. The nature of a state $\mathbf{u}_s$ varies according to the model that is chosen to represent the parametrisation. For example, when using a 2D parametrisation that represents translations, $\mathbf{u}_s$ contains two offsets: dx and dy. However this can also be expressed with matrix notation, as we just explained. Therefore, for a state $\mathbf{u}_s$ containing the coefficients of a transformation matrix, we call U_s the matrix built from these coefficients, and $U_s * \mathbf{s}$ the coordinate of the 2D pixel transformed with the matrix U_s, $I(U_s * \mathbf{s})$ is the value of image I at the transformed pixel position.

5.4.2 Methodology

Model

In Chapter 4, the transformation model used in the stereo approach uses planes in the horizontal disparity space, and can be defined using a partial affine transformation:

$$U'_s = \begin{pmatrix} 1+a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{5.15}$$

which yields, when applied to a pixel $\mathbf{s}$, an offset of $U_s * \mathbf{s} = [ax_s + by_s + c, 0]^\intercal$, corresponding to the model seen in Chapter 4, where the matrix U_s is defined by $U_s = U'_s - I$.

We propose to parameterise the flow field with affine transformations, so that a state at a node s corresponds to the transformation:

$$U'_s = \begin{pmatrix} 1+a & b & c \\ d & 1+e & f \\ 0 & 0 & 1 \end{pmatrix}, \tag{5.16}$$

which yields a final 2D displacement of $U_s * \mathbf{s} = [ax_s + by_s + c, dx_s + ey_s + f]^\intercal$ with $U_s = U'_s - I$. Given the form of the final displacement, this can be seen as the formulation previously presented in Chapter 4 extended to optical flow. Geometrically, it allows more variation than the model described above in section 5.3, as it includes shearing in addition to rotation and scaling.

A straightforward way of defining the states would be to simple use the 6 coefficients of the affine transformation. However, we are looking for a convenient way of randomising around a given affine transformation, which is not trivial when dealing directly with these coefficients. We use a similar method to the technique described in Chapter 4, which includes a representation of the problem in four dimensions. We consider the disparity space where the disparity is defined as a 2D element $[\delta_x, \delta_y]$. This space is a 4-dimensional space, where the dimensions are $[x, y, \delta_x, \delta_y]$. We are looking for a way of representing a 3D plane in this space, so that it can be mapped to the 3D surface of the physical world. We can do so by intersecting two 4D planes. Let us consider the two following non parallel 4D planes: Π and Π' whose normals are $\mathbf{n} = [n_x, n_y, n_{\delta_x}, 0]$ and $\mathbf{n}' = [n'_x, n'_y, 0, n'_{\delta_y}]$ respectively. Let us now consider a point $p_0 = [x_0, y_0, \delta_{x0}, \delta_{y0}]$ located on both planes and therefore verifying the two following equations:

$$n_x x_0 + n_y y_0 + n_{\delta_x}\delta_{x0} + C = 0 \tag{5.17}$$

$$n'_x x_0 + n'_y y_0 + n'_{\delta_y}\delta_{y0} + C' = 0 \tag{5.18}$$

The first plane establishes a relation between the 2D coordinates of a point in the image $[x_0, y_0]$, and its disparity in the x direction in the second image. The second plane does the same thing but with the disparity in the y direction. Since p_0 is located on both planes, it means that it is on the intersection of the two 4D planes, which means that it is located on a 3D plane, that can actually be mapped to the 3D surface in the physical world, where p_0 lies. Let us now consider a point $p = [x, y, \delta_x, \delta_y]$ and assume that p is located on the same 3D surface as p_0. Then, it must also follow the equations of both planes:

$$n_x x + n_y y + n_{\delta_x}\delta_{x0} + C = 0 \tag{5.19}$$

$$n'_x x + n'_y y + n'_{\delta_y}\delta_{y0} + C' = 0 \tag{5.20}$$

which means that we also have:

$$n_x x + n_y y + n_{\delta_x}\delta_x = n_x x_0 + n_y y_0 + n_{\delta_x}\delta_{x0} \tag{5.21}$$

$$n'_x x + n'_y y + n'_{\delta_y}\delta_y = n'_x x_0 + n'_y y_0 + n'_{\delta_y}\delta_{y0} \tag{5.22}$$

and thus:

$$\delta_x = -\frac{n_x}{n_{\delta_x}}x - \frac{n_y}{n_{\delta_x}}y + \frac{n_x x_0 + n_y y_0 + n_{\delta_x}\delta_{x0}}{n_{\delta_x}} \tag{5.23}$$

$$\delta_y = -\frac{n'_x}{n'_{\delta_y}}x - \frac{n'_y}{n'_{\delta_y}}y + \frac{n'_x x_0 + n'_y y_0 + n'_{\delta_y}\delta_{y0}}{n'_{\delta_y}} \tag{5.24}$$

which, by defining a, b, c, d, e and f as:

$$\begin{aligned} a &= -\frac{n_x}{n_{\delta_x}}, \quad b = -\frac{n_y}{n_{\delta_x}}, \quad c = \frac{n_x x_s + n_y y_s + n_{\delta_x}\delta_{x0}}{n_{\delta_x}} \\ d &= -\frac{n'_x}{n'_{\delta_y}}, \quad e = -\frac{n'_y}{n'_{\delta_y}}, \quad f = \frac{n'_x x_s + n'_y y_s + n'_{\delta_y}\delta_{y0}}{n'_{\delta_y}} \end{aligned} \tag{5.25}$$

corresponds to the displacement given by the affine transformation from equation 5.4.2. This provide us with a way of generating the coefficients of the matrix, by using 2 unit vectors and 2 disparity values, in x and y. Furthermore, randomisation around a given state becomes similar to that of Chapter 4, as we can simply jitter around the disparity and the normal of the plane.

This is a direct extension of the method of Chapter 4, which means that, if the scene is static, planes in the first view are mapped to a plane in the other view if the images are rectified. As we know, in a standard optical flow setup objects are moving, and it is impossible to rectify the images. This is why we later incorporate 2 more degrees of freedom which give us a homography, to be able to represent accurately the displacement in the world. This is the method presented in Chapter 6. However, in this section we use an affine transformation, which can be seen as an approximation, as we assume that the images are rectified, while they are not. An advantage is that it incorporates shearing and therefore can capture more complex transformations than the model defined in section 5.3.

Mechanisms

Our algorithm PMBP requires the implementation of three fundamental mechanisms: random initialisation, random resampling and propagation.

Initialisation For the initialisation step we need a way of generating random states. In general, as the dimensionality of the state space increases, it becomes more difficult to sample states that will produce a 2D displacement that gives a match within the boundaries of the image. However, for the affine transformation, it is fairly straightforward to generate. Using the formulation mentioned above, we compute the coefficients of the matrix as mentioned in equation 5.4.2, and thus only need to generate 2 random unit vectors and 2 random disparities in x and y within the range of allowed motion to generate all the required coefficients. Using these coefficients ensures that the displacement at pixel s is equal to $U_s * [x_s, y_s]^\intercal = [\delta_x, \delta_y]^\intercal$, and produces an affine warping for pixels in the patch around s. In terms of the implementation, and similarly to Chapter 4, it is useful to keep the parameters $[n_x, n_y, n_z, \delta_x, n'_x, n'_y, n'_z, \delta_y]$ stored in memory as it makes the randomisation and the propagation much easier to directly jitter the values of the normals and the displacements in an intuitive way.

Randomisation For this step we need to sample a new affine transformation at a given distance from another given transformation. We use a method similar to that of Chapter 4. We sample two disparity differential values Δ_{δ_x} and Δ_{δ_y} from the intervals $[-\Delta_\delta^{\max}, \Delta_\delta^{\max}]$. and two differential vectors, Δ_n

and Δ'_n, from the intervals $[-\Delta_n^{\max}, \Delta_n^{\max}]$ and $[-\Delta_{n'}^{\max}, \Delta_{n'}^{\max}]$ respectively. As previously, the sizes of these intervals vary according to the standard concentric circle sampling mechanism of PatchMatch. After having retrieved these random differential values, we update the final components $\delta_x^* = \delta_x + \Delta_{\delta_x}$, $\delta_y^* = \delta_y + \Delta_{\delta_y}$, $n^* = n + \Delta_n$ and $n'^* = n' + \Delta'_n$, and rebuild the affine transformation following the methodology explained in the previous section.

Propagation Propagating a state u_t from node t to node s requires some processing on u_t, in order to generate a state u_s suitable for s. Given that we keep in memory the extra parametrisation explained in the initialisation section, we have $u_t = [n_x, n_y, n_z, \delta_x, n'_x, n'_y, n'_z, \delta_y]$ and $u_s = [m_x, m_y, m_z, \gamma_x, m'_x, m'_y, m'_z, \gamma_y]$ such that:

$$[m_x, m_y, m_z] = [n_x, n_y, n_z] \tag{5.26}$$

$$[m'_x, m'_y, m'_z] = [n'_x, n'_y, n'_z] \tag{5.27}$$

$$\gamma_x = -\frac{n_x}{n_z}x_t - \frac{n_y}{n_z}y_t + \frac{n_x x_t + n_y y_t + n_z z}{n_z} \tag{5.28}$$

$$\gamma_y = -\frac{n'_x}{n'_z}x_t - \frac{n'_y}{n'_z}y_t + \frac{n'_x x_t + n'_y y_t + n'_z z}{n'_z} \tag{5.29}$$

This gives us a new candidate for a neighbouring node, that takes into account the local affine transformation of the patch.

5.4.3 Experiments

To illustrate the benefits of using the affine model over the 5D model presented previously in section 5.3, we use a pair of images that include a plane motion with significant deformation due to the perspective projection. While the affine model is not able to capture exactly this transformation, it produces significantly better results than the 5D model. The resulting flow fields can be seen in figure 5.18 as well as the ground truth.

Similarly to Chapter 4 and section 5.3, we run our algorithm with varying the regularisation level on one case and plot the evolution of the EPE. The remaining parameters are fixed and chosen experimentally. Results can be seen in figure 5.19, where we observe that the regularisation is beneficial to the resulting flow.

Finally, we apply our method with and without regularisation term to the UCL dataset [Mac Aodha et al., 2010] and the Middlebury training set [Baker et al., 2011] for which we have access to the ground truth flow and therefore we can use it to provide quantitative results. Note that all our experiments in this section have been run with the same parameters as in section 5.3.2. The results are summarised in table 5.2. We observe that for 27 out of 32 cases, the model including the regularisation term performs as well or better than the model without the smoothness term.

5.4.4 Discussion

As in the previous experiments, the use of a smoothness term is proven, and the affine model performs well on cases were the deformation of the patches cannot be explained by translation, rotation and scaling only. Naturally, the affine model is still limited as it is not able to recover all the possible plane motions in the scene, which motivates the use of a full homography model that we present in Chapter 6.

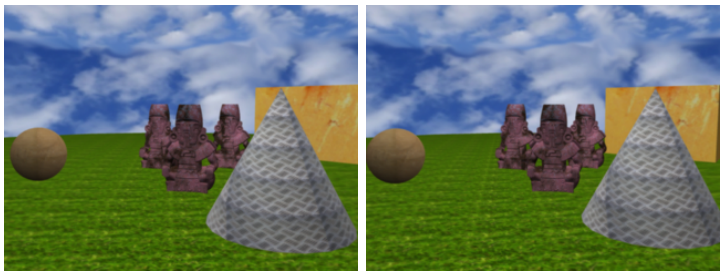
	6D PM	6D PMBP
UCL Lg. Disp.		
Crates1	4.35	**3.87**
Crates2	**2.627**	3.07
Mayan1	0.219	**0.197**
Robot	**2.183**	2.25
Crates1Htxtr2	0.423	**0.339**
Crates2Htxtr1	0.92	**0.858**
Brickbox1t1	0.229	**0.228**
Brickbox2t2	**0.393**	0.407
GrassSky0	0.29	**0.263**
GrassSky9	0.336	**0.269**
blow19Txtr2	0.218	**0.203**
drop9Txtr2	**1.341**	1.361
street1Txtr1	6.554	**6.47**
UCL Sm. Disp.		
Mayan2	0.225	**0.197**
YosemiteSun	0.817	**0.642**
GroveSun	0.319	**0.309**
Sponza1	2.365	**2.36**
Sponza2	**2.51**	**2.51**
TxtRMovement	**1.26**	**1.26**
TxtLMovement	1.32	**1.24**
blow1Txtr1	0.054	**0.049**
drop1Txtr1	0.075	**0.07**
roll1Txtr1	**0.003**	**0.003**
roll9Txtr2	0.02	**0.019**

(a)

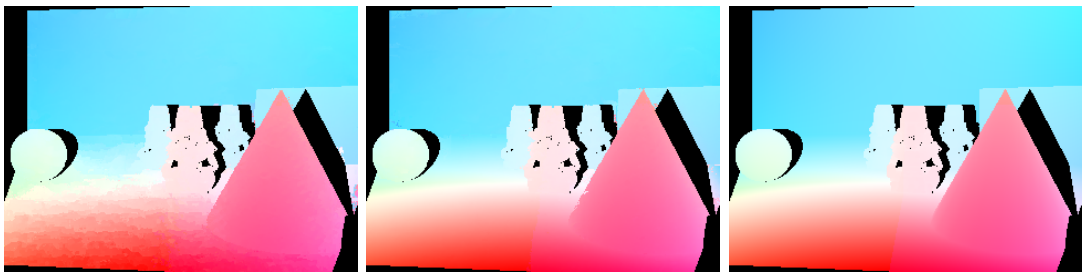
	6D PM	6D PMBP
Middlebury		
Venus	0.348	**0.3334**
Urban3	**1.159**	1.162
Urban2	0.33	**0.327**
RubberWhale	0.136	**0.135**
Hydrangea	0.202	**0.199**
Grove3	0.577	**0.568**
Grove2	0.226	**0.218**
Dimetrodon	0.191	**0.186**

(b)

Table 5.2: Comparison of EPE output from the two Affine methods without (6D PM) and with (6D PMBP) regularisation term, on (a) the UCL data set and (b) the Middlebury dataset. Bold values indicate the best performance.



(a) Image 1 (b) Image 2



(c) 5D model result, EPE = 0.437 (d) Affine model result, EPE = 0.197 (e) Ground Truth

Figure 5.18: Comparison of the flow produced by the 5D model, shown in (c), with the flow produced by the affine model shown in (d). The original images are shown in (a) and (b). The ground truth is shown in (e). We see that the 5D model struggles at capturing the flow on the ground plane which undergoes a perspective transformation as the camera moves. The affine model however, produces a result that is much closer to the ground truth solution. Note that this figure is best viewed in the electronic copy of this thesis.

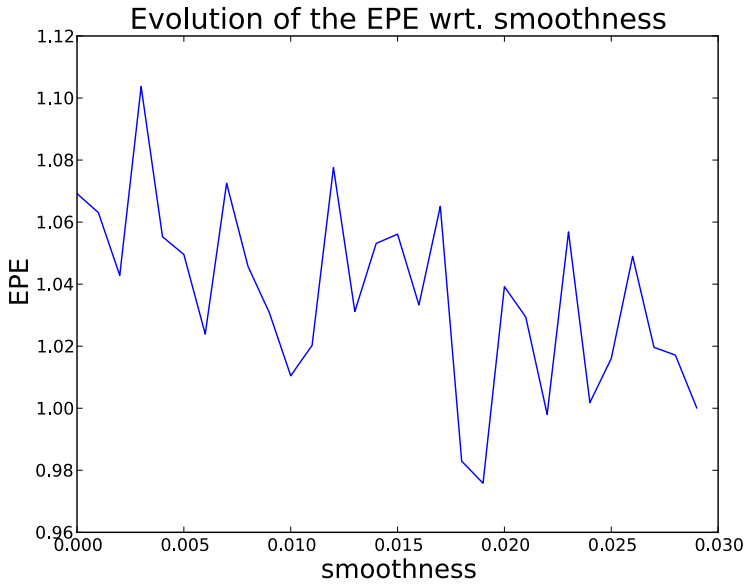
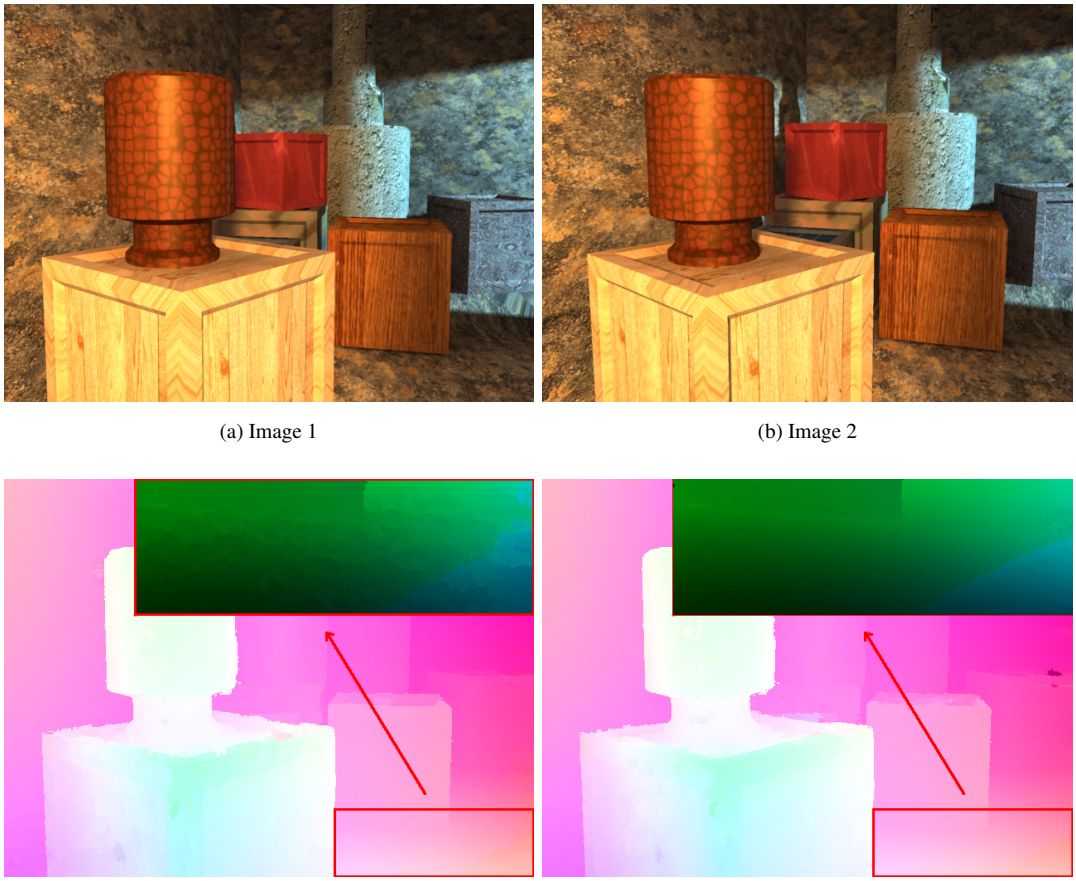


Figure 5.19: Evolution of the End Point Error (EPE) with respect to the regularisation coefficient.

5.5 Comparison and Discussion



(a) Image 1 (b) Image 2

(c) 5D result on 'Crates1Htxtr2' (d) 6D result on 'Crates1Htxtr2'

Figure 5.20: Comparison of the results when using the 5D and the 6D affine model on a case displaying significant deformation due to perspective distortion. The original images are displayed in (a) and (b). (c) and (d) illustrate the results for the 5D and the 6D affine model respectively. An area of the image has been enlarged and inverted to better show the details of the resulting flow. We can see that while the 5D model produces a piecewise constant flow, the 6D model outputs a much smoother flow as it is able to better capture the perspective distortion. Note that this figure is best viewed in the electronic copy of this thesis.

In this section we summarise the results of all three different techniques presented to far on the two datasets (UCL and Middlebury) in order to establish a comparison. Note that all the experiments are run with the same parameters such as the pairwise term weight and truncation value to facilitate their comparison. As most of these parameters were also used in the previous chapter on different datasets, we believe they could be used to process entirely new datasets successfuly. Results can be seen in figures 5.21, 5.22 and 5.23 . We can make several observation from these outputs. First, in both models the smoothness term included in PMBP is beneficial to the quality of the flow output. We show a visual comparison for the affine model of the effect of the smoothness term in figure 5.24. This comes in agreement with the results of the stereo matching application presented in Chapter 4 which show the

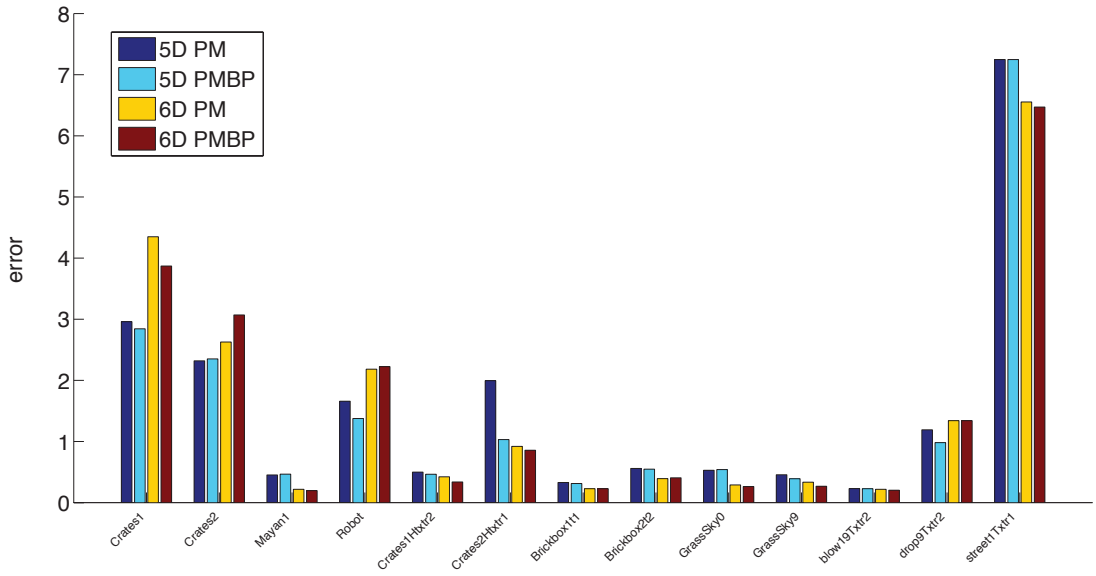


Figure 5.21: Comparison of the four methods on the large displacement cases of the UCL dataset.

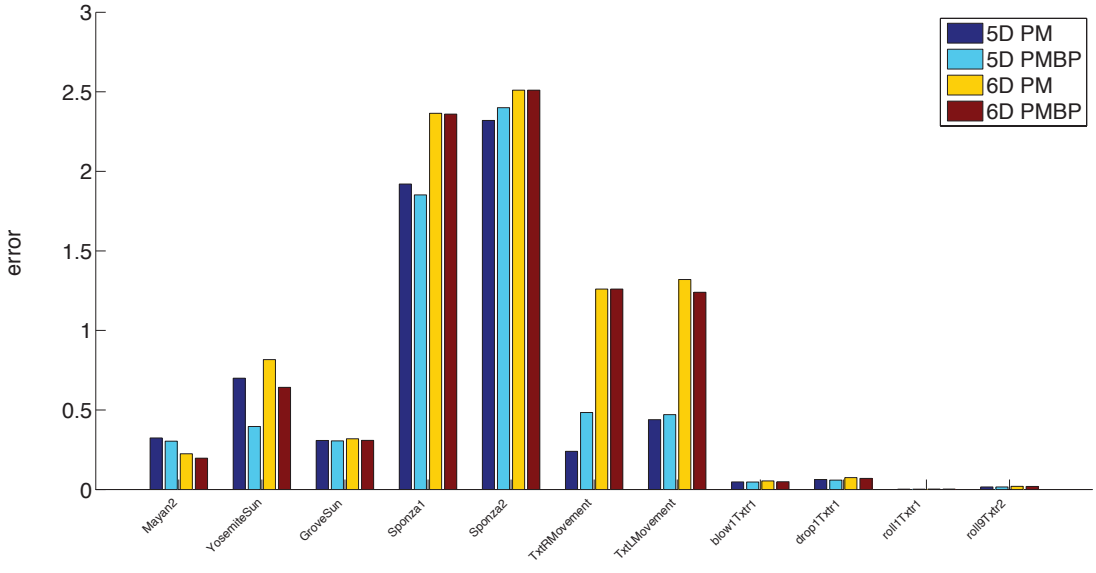


Figure 5.22: Comparison of the four methods on the small displacement cases of the UCL dataset.

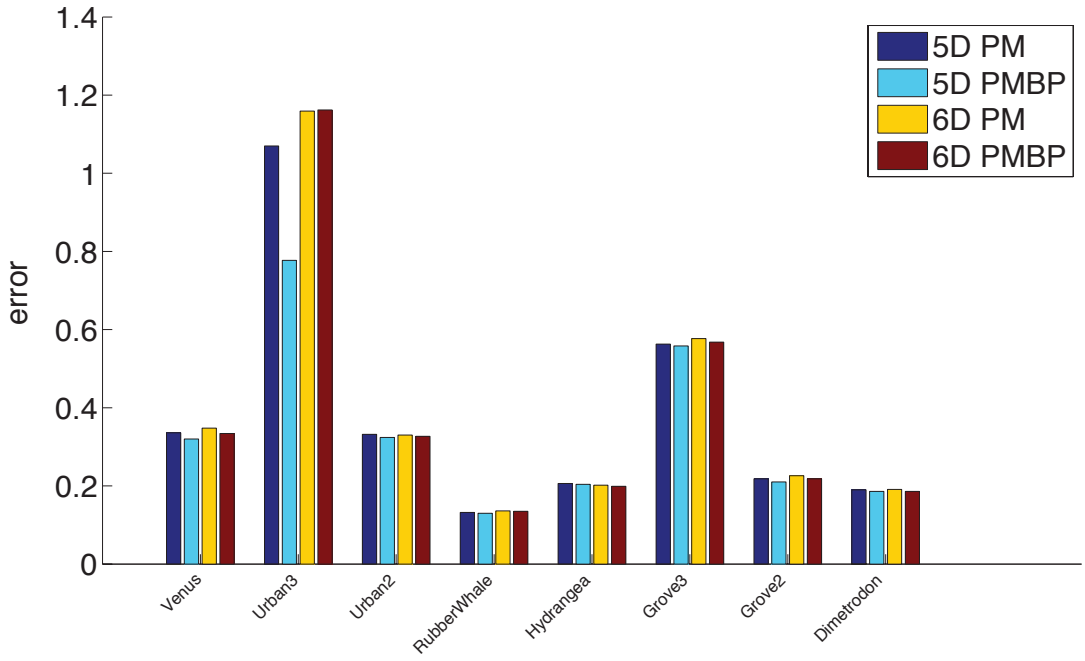
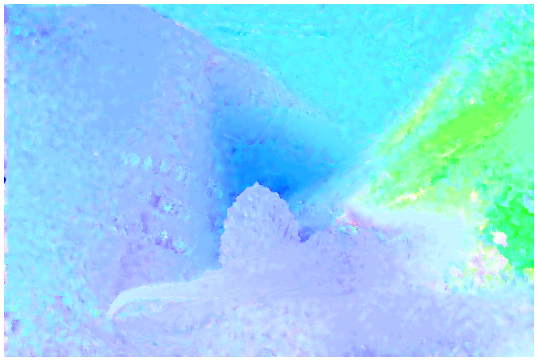


Figure 5.23: Comparison of the four methods on the Middlebury dataset.

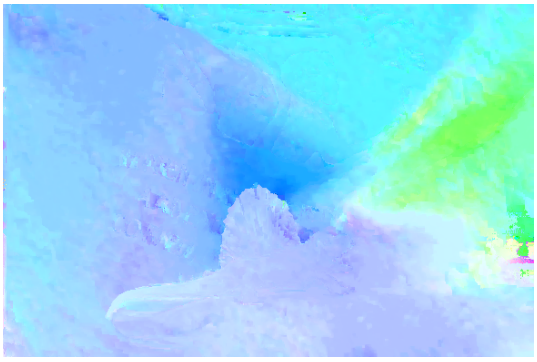
same phenomenon. Then, it seems that the 5D model produces slightly better results overall than the affine. Indeed, in 21 out of 32 image pairs the 5D model performs better. This might be surprising at first, as the affine model was designed to capture better the change in appearance induced in slanted planes. However, there are some images pairs where the affine model performs significantly better than the 5D model. Such image pairs and their results can be seen in figures 5.18 and 5.20. These images seem to have a type of transformation that involves large amount of perspective distortion, which the affine model can handle in a better way. The fact that the 5D model perform slightly better on the remaining cases can explained by the type of deformation present in these image pairs, which can be captured adequately by the model. Therefore adding 1 degree of freedom (with the affine model) increases substantially the search space, which leads to a loss of quality. This phenomenon has been already observed in Chapter 4, where we conducted an experiment and saw that an increase in the range of the search space also increases the uncertainty and leads to a loss of quality. We can conclude that the model used for optical flow, in our framework, needs to be complex enough to capture all possible deformations, but with as few degrees of freedom as possible, in order to ensure robustness. This is a reasonable assumption when considering that our framework comprises a large randomisation component, and having a smaller search space ensures that more random "guesses" will be correct, which then accelerates the optimisation process.

(a) Image 1

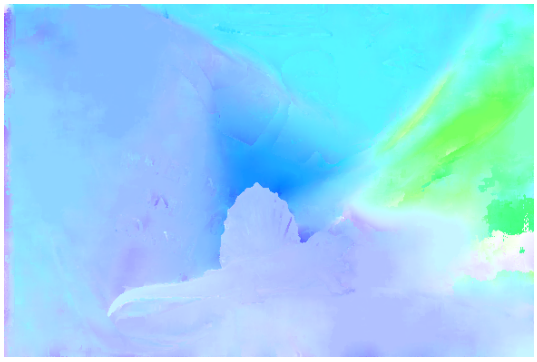
(b) Image 2



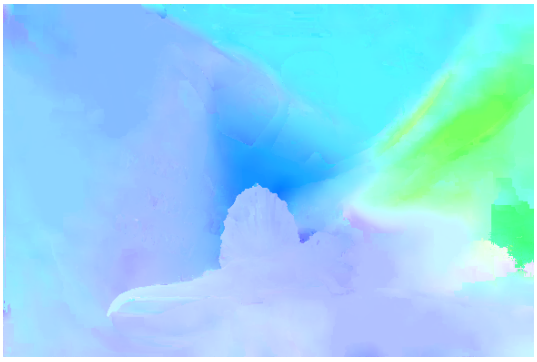
(c) Affine result, 7x7, smoothness term disabled



(d) Affine result, 7x7, smoothness term enabled



(e) Affine result, 21x21, smoothness term disabled



(f) Affine result, 21x21, smoothness term enabled

Figure 5.24: Effect of the smoothness term on the “Dimetrodon” case. (a) and (b) show the original images, (c) and (e) show the result with no smoothness term for a patch size of 7x7 and 21x21 respectively, and (d) and (f) show the result with smoothness term enabled for these patch sizes.

Chapter 6

A Geometrical Model for Optical Flow

In algorithms that use patches to address the optical flow problem, the patch size becomes an important algorithm parameter: a small window offers little robustness to intensity variations such as those caused by lighting change, differences in camera response, or image noise; a large window can overcome these difficulties but most published work suffers from what we loosely term the "fronto-parallel" assumption, according to which each point in the window is assumed to be undergoing the same 2D translation.

The robustness of small-window models can be improved by the use of priors relating the motion at neighbouring pixels, but first-order priors themselves typically imply the fronto-parallel (FP) limitation, second-order priors are expensive to optimize for general energies [Woodford et al., 2009] although efficient schemes exist for some cases [Trobin et al., 2008]. Beyond second order, higher-order priors impose quite severe limitations on the state spaces they can model. In the case of optical flow, the state space is essentially continuous, and certainly any discretisation must be very dense.

An alternative strategy to relax the FP assumption is to *overparameterise* the motion field. Instead of modelling the correspondence field as a two degree of freedom (2-dof) translation at each pixel, previous work has used a 3-dof similarity transformation [Barnes et al., 2010], 6-dof affine transformation [Nir et al., 2008] or 6-dof linearised 3D motion model [Nir et al., 2008]. We have also shown a 5D and a 6D parameterisation in the previous chapter that use PMBP. With such models, even first order priors can be expressive (e.g., piecewise constant surface normal is equivalent to piecewise constant depth derivatives rather than piecewise constant depth).

In this chapter, we apply a new overparameterisation to the computation of optical flow, using a full homography at each pixel. We show that PMBP can be adapted to optimise this model effectively, and that the resulting flow fields compare favourably to the state of the art on a number of datasets. The model parameterises a separate plane under rigid body motion at each pixel, and the prior can be set to specialise the model for piecewise rigid motion, or indeed for a stereo scene, reducing to the existing PMBP stereo algorithm. The algorithm is also easily adapted to impose a global fundamental matrix prior [Valgaerts et al., 2008, Wedel et al., 2008].

6.1 Cameras and homographies

It has been shown that two images of the same planar surface can be related by a unique homography, assuming a pinhole camera model [Hartley and Zisserman, 2000]. This principle is key to the new optical

flow model that we propose in this section.

Before we introduce our new optical flow model, we describe the projection model of a pinhole camera and how it can be represented by a homography. A pinhole camera is fully characterised by two matrices. The first is the intrinsic matrix, which characterises the focal length, the image format and the principal point. It is usually referred to as K and is defined as:

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.1}$$

where $[\alpha_x, \alpha_y] = [fm_x, fm_y]$ is the focal length in pixels, f is the focal length in physical distance, $[m_x, m_y]$ are scaling factors in pixel per meter, γ is the skew coefficient, and $[u_0, v_0]$ are the coordinates of the principal point of the camera, which is usually in the centre of the image.

The second matrix that characterises a camera is its extrinsic matrix. It refers to the position and orientation of the camera in the physical space. It is usually broken down into two elements: a rotation matrix R and a translation vector $\mathbf{t}$, such that:

$$M = [\, R \mid \mathbf{t} \,] = \begin{bmatrix} \omega_{00} & \omega_{01} & \omega_{02} & \tau_x \\ \omega_{10} & \omega_{11} & \omega_{12} & \tau_y \\ \omega_{20} & \omega_{21} & \omega_{22} & \tau_z \end{bmatrix}. \tag{6.2}$$

We can relate a point in physical space $X = [x, y, z, 1]^\top$ with its projection in image space $p = [u, v, 1]^\top$ as follows:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{00} & \omega_{01} & \omega_{02} & \tau_x \\ \omega_{10} & \omega_{11} & \omega_{12} & \tau_y \\ \omega_{20} & \omega_{21} & \omega_{22} & \tau_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{6.3}$$

where λ is a scaling factor.

Now, let us consider a plane in the scene. We assume that for a point q on this plane, there exists a coordinate system in which it can be expressed with 2 coordinates only. Therefore, in homogeneous coordinates it can be written as:

$$q = \begin{bmatrix} u' \\ v' \\ 0 \\ 1 \end{bmatrix}. \tag{6.4}$$

When using this point in equation 6.3, we get:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{00} & \omega_{01} & \omega_{02} & \tau_x \\ \omega_{10} & \omega_{11} & \omega_{12} & \tau_y \\ \omega_{20} & \omega_{21} & \omega_{22} & \tau_z \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 0 \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{00} & \omega_{01} & \tau_x \\ \omega_{10} & \omega_{11} & \tau_y \\ \omega_{20} & \omega_{21} & \tau_z \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$
$$= \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = H \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}, \tag{6.5}$$

where H is a homography. This means that the motion of the pixels of a moving plane can be characterised by a unique homography (the camera can be moving as well - it is equivalent). This property lies at the core of our method as we will see below.

6.2 Method

Let (I_1, I_2) be an ordered pair of images depicting a static or moving scene at different points in time and/or from different points of view, each consisting of a total of n pixels. Let $\mathbf{x}_s = (x_s, y_s)^\top$ denote the position of pixel s, such that $s \in \{1, \ldots, n\}$. Moreover, let $N(s)$ be the set of indices of the 4-connected neighbours of the pixel $\mathbf{x}_s$, and $W(s)$ the set of indices of pixels in the $P \times P$ patch around $\mathbf{x}_s$. At every pixel $\mathbf{x}_s$, rather than seeking a 2D flow vector, we aim to obtain a parameter vector $\mathbf{u}_s$ that defines a 3×3 homography $H(\mathbf{u}_s) = H_s$ describing the motion of points near $\mathbf{x}_s$. We solve for the flow field by minimizing an energy defined over such parameter vectors, and comprising *data terms* ψ_s and *smoothness terms* ψ_{st}:

$$E(\mathbf{u}_1, ..., \mathbf{u}_n) = \sum_{s=1}^{n} \psi_s(H(\mathbf{u}_s)) + \lambda \sum_{s=1}^{n} \sum_{t \in N(s)} \psi_{st}(\mathbf{u}_s, \mathbf{u}_t). \tag{6.6}$$

In the following section, we proceed to introduce the parameterisation and the data term, followed by a discussion of the smoothness term in the section thereafter.

6.2.1 Model and data term

Let $I_i(\mathbf{x}_s) \in \mathbb{R}^4$ denote, at pixel $\mathbf{x}_s$ in image i, the colour vector augmented with gradient magnitude. For $\mathbf{x}$ at 2D floating point coordinates, we obtain $I(\mathbf{x})$ using bilinear interpolation. Let $\mathring{\mathbf{x}} = (x_1, x_2, x_3)^\top \in \mathbb{P}^2$ denote a pixel in projective 2-space; we denote its analogue in Euclidean 2-space as $\epsilon(\mathring{\mathbf{x}}) = (x_1/x_3, x_2/x_3)^\top \in \mathbb{R}^2$. Let $H * \mathbf{x} = \epsilon(H(\mathbf{x}^\top, 1)^\top)$ denote the application of a homography H to the pixel $\mathbf{x}$. Given a homography H_s at the pixel $\mathbf{x}_s$ in image i, the brightness constancy assumption becomes $I_i(\mathbf{x}_s) = I_j(H_s * \mathbf{x}_s)$, where $i, j \in \{1, 2\}, i \neq j$. This translates to a data term that, at the pixel $\mathbf{x}_s$, sums over the pixels of the patch $W(s)$:

$$\psi_s(H_s) = \sum_{t \in W(s)} w_{st} \| I_1(\mathbf{x}_t) - I_2(H_s * \mathbf{x}_t) \|_\theta / |W(s)|, \tag{6.7}$$

where the norm $\left\|\right\|_\theta$ is the one defined in equation 4.2.1, which we recall as being:

$$\left\|\hat{v}\right\|_\theta = \alpha_c \min\left(\|[v_1, v_2, v_3]^\intercal\|, \tau_c\right) + \alpha_\nabla \min\left(v_4, \tau_\nabla\right) \tag{6.8}$$

for any 4 dimensional vector $\hat{v} = [v_1, v_2, v_3, v_4]^\intercal$ with weights $(\alpha_c, \alpha_\nabla)$ and truncation terms (τ_c, τ_∇). The weights w_{st} are adaptive support weights [Yoon and Kweon, 2006] which we used in the previous chapters, defined by

$$w_{st} = \exp(- \parallel I(x_s, y_s) - I(x_t, y_t) \parallel /\omega), \tag{6.9}$$

where ω controls the influence of the adaptive weighting. In our model, we effectively compare a square patch in the first image with a patch warped under the candidate homography in the second image.

In our method, we parameterise not only a scene plane at each pixel, but also a rigid body motion transforming points to their appropriate respective positions in the second image, which we subsequently project into image space. For this derivation, we shall assume a known 3×3 camera calibration matrix K, although we have not found the algorithm to be particularly sensitive to its accuracy in our experiments.

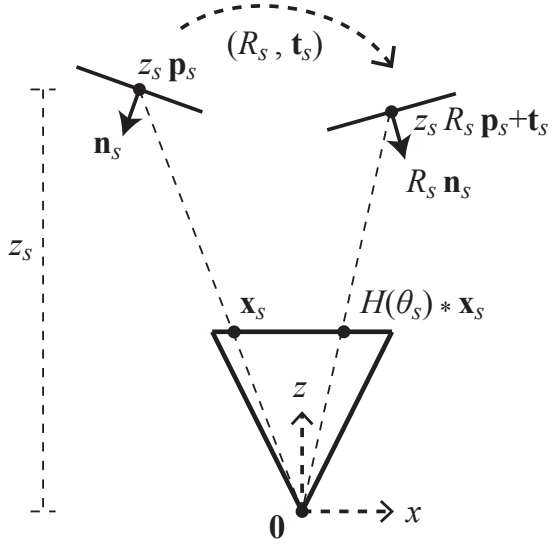


Figure 6.1: Representation of the transformation at pixel $\mathbf{x}_s$.

Let $\mathbf{p}_s = K^{-1}(\mathbf{x}_s^\top, 1)^\top \in \mathbb{R}^3$ denote the back-projection of $\mathbf{x}_s$ and $z_s \in \mathbb{R}^+$ the scene depth in the first image, so that the pre-image of $\mathbf{x}_s$ is the point $z_s\mathbf{p}_s$. Let $\mathbf{n}_s \in \mathbb{R}^3$ be the surface normal at $z_s\mathbf{p}_s$. Ignoring for the moment the details of the parameterisation, we write our overparameterised motion model as

$$H(\mathbf{u}_s) = K\left(R_s + \frac{1}{z_s\mathbf{n}_s^\top\mathbf{p}_s}\mathbf{t}_s\mathbf{n}_s^\top\right)K^{-1}, \tag{6.10}$$

where $\mathbf{u}_s = (z_s, \mathbf{n}_s, R_s, \mathbf{t}_s)^\top$ for a total of 9-dof. Setting $z_s\mathbf{n}_s^\top\mathbf{p}_s = -d_s$, we obtain the familiar *homography induced by the plane* (see [Hartley and Zisserman, 2000]), with a plane $\pi_s = (\mathbf{n}_s^\top, d_s)^\top \in \mathbb{P}^3$ and a camera $K[R_s \mid \mathbf{t}_s]$ expressed in the camera coordinate frame of the camera of image 1, likewise

with calibration matrix K. For static scenes undergoing only camera motion, the camera $K[R_s \mid \mathbf{t}_s]$ corresponds to the camera of image 2. More generally, this homography lends itself to interpretation as the rigid body motion $g_s = (R_s, \mathbf{t}_s) \in SE(3)$ applied—with camera pose fixed—to the points obtained by intersecting pixel back-projections from image 1 with π_s, yielding, upon reprojection to image space, the corresponding pixels in image 2 (cf. Figure 6.1). Note that on this interpretation, we may treat pure camera motion, pure object motion, and joint camera and object motion in the same framework. A plane whose normal does not point towards the camera, however, is physically impossible, and one that is nearly orthogonal to the given look direction vector is of no practical use in obtaining matches. Accordingly, for a homography $H(\mathbf{u}_s)$ to be deemed *valid*, we additionally require that the normals $\mathbf{n}_s$ and $R_s\mathbf{n}_s$ are facing toward the camera and are within 85° of the look direction vectors corresponding to $z_s\mathbf{p}_s$ and $z_sR_s\mathbf{p}_s + \mathbf{t}_s$, respectively. We assign to invalid homographies an infinite cost.

6.2.2 Smoothness term

The pairwise energy ψ_{st} encourages the states $\mathbf{u}_s$ and $\mathbf{u}_t$ to be similar. There are several strategies that can be applied in order to build this pairwise term. The natural choice is to define distances between the geometric quantities at each pixel, for example angles between normals, rotation axes and translation directions, as well as magnitude differences between depths and rotation angles. For example, one can define, with $\rho(\tau, t)$ some robust function such as $\min(t^2, \tau)$:

$$\begin{aligned}\psi_{st}^{\text{geom}}(\mathbf{u}_s, \mathbf{u}_t) :=& \lambda_1\rho(\tau_1, z_s - z_t) + \\ & \lambda_2\rho(\tau_2, \cos^{-1}(\mathbf{n}_s^\intercal\mathbf{n}_t)) + \\ & \lambda_3\rho(\tau_3, \cos^{-1}(\text{axis}(R_s)^\intercal\text{axis}(R_t))) + \\ & \lambda_4\rho(\tau_4, \text{angle}(R_s) - \text{angle}(R_t)) + \\ & \lambda_5\rho(\tau_5, \cos^{-1}(\mathbf{t}_s^\intercal\mathbf{t}_t)).\end{aligned} \tag{6.11}$$

where $[\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5]$ are weights, $[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5]$ are truncation terms, z_s and z_t are the depth components of $\mathbf{u}_s$ and $\mathbf{u}_t$, $\mathbf{n}_s$ and $\mathbf{n}_t$ are the plane normal components, R_s and R_t the rotation matrices, and $\mathbf{t}_s$ and $\mathbf{t}_t$ are the translation vectors.

However, such a definition introduces a large number of algorithm tuning parameters, as the natural scales of variation of each parameter type are not commensurate. These could be determined on a training set, but a large training set may be required, which is potentially impractical to obtain for real-world applications.

An alternative simple measure exists, which is computed between the homographies H_s and H_t by measuring the difference in the effect they have on the four corners of a patch. Let $C(s)$ be defined as the set of indices of the four corner pixels of the patch centred at $\mathbf{p}_s$. The difference is then defined as

$$\psi_{st}(\mathbf{u}_s, \mathbf{u}_t) = w_{st} \sum_{c \in C(s)} \min\big(\kappa, \|H(\mathbf{u}_s) * \mathbf{p}_c - H(\mathbf{u}_t) * \mathbf{p}_c\|\big) \tag{6.12}$$

where κ is a truncation cost and w_{st} is defined in equation 6.9. This difference has natural units of pixels, and means that the smoothness term now has only two parameters (λ and κ), as compared to the ten of equation (6.11). The smoothness term is illustrated in figure 6.2.

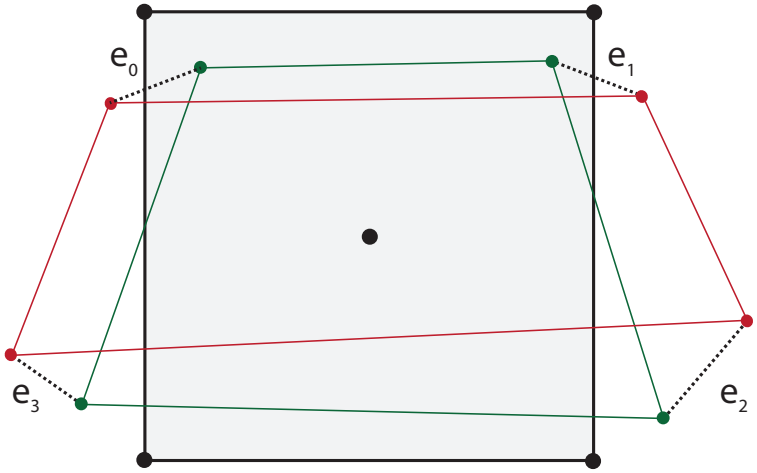


Figure 6.2: Four point pairwise term. The original patch is shown in black. The red points show the original points are transformed with the first state u_t, while the green points show the original points transformed with the second state u_s. The difference in point positions represent the four errors e_0, e_1, e_2, and e_3 which are truncated and added.

6.2.3 Minimisation

It may be easy to formulate a realistic energy function, but such a function is of little practical use if it cannot be optimized in reasonable time. Minimizing the energy presented above reduces to a nonconvex optimisation problem over a high-dimensional, continuous state space. PMBP provides an avenue to optimizing over such a space by combining PatchMatch with Belief Propagation. Although PatchMatch on its own is, strictly speaking, only a minimizer of the unary energy—adopting, for each pixel and at each iteration, a new parameter vector if doing so yields a lower unary cost—it can exploit the eventuality of underlying spatial coherence of a parameter space by attempting to propagate states to neighbouring pixels, provided such propagation is sensible. PMBP additionally allows for promoting smoothness of the resulting parameter space explicitly, by making use of pairwise terms.

We adapt PMBP in the aim of assigning to each pixel $\mathbf{x}_s$ an optimal state $\mathbf{u}_s$, mapping the $P \times P$ patch centered on $\mathbf{x}_s$ in one image to the corresponding warped patch in the other. The steps that we describe next correspond to our additions and modifications over the original PMBP algorithm. We begin by (i) initializing the parameter space in a semi-random manner. Next, for i iterations and at each pixel $\mathbf{x}_s$, we (ii) attempt to propagate states from neighbours of $\mathbf{x}_s$, (iii) try perturbing the current parameter vector randomly (random search), and (iv) applying a local RANSAC plane fit to neighbouring reconstructed points. If we are computing the flow in both directions, we (v) additionally attempt to propagate the state at $\mathbf{x}_s$ from the given image to $H_s * \mathbf{x}_s$ in the other. Some of these steps are part of a joint work, and in the next paragraphs we explicitly mention when this is the case. These steps are described in detail in Appendix A and [Besse et al., 2013a].

(i) Initialisation. We estimate the dominant rigid body motion of the scene using RANSAC on SIFT feature matches to recover the essential matrix $E = [\mathbf{t}_E]_\times R_E$, which we decompose by means of standard techniques [Hartley and Zisserman, 2000]. In static scenes undergoing only rigid camera motion, the rigid body motion $g_E = (R_E, \mathbf{t}_E) \in SE(3)$ expresses the camera $K[R_E \mid \mathbf{t}_E]$ of image 2 in the camera coordinate frame of the camera of image 1. We initialise the rotation and translation components of each state using $R_E, \mathbf{t}_E$. The normals are randomly sampled within the hemisphere facing the camera, and the inverse of the depths are uniformly sampled within an interval defined by the depths of the triangulated SIFT points. We run the full algorithm in a coarse-to-fine scheme to initialise the solution for the full resolution images.

(ii) Propagation. Within PMBP, there is the freedom to choose a so-called message-passing schedule. We use a standard scheduling which consists in sweeping the image along the four main directions (top-left to bottom-right, top-right to bottom-left, bottom-right to top-left, bottom-left to top-right) and accordingly run the algorithm for 4 iterations, in order to get one sweep along each main direction. For a given pixel, we use the particles of its neighbours as a source of new candidates in the same fashion as PMBP. We also combine the current particles with components of the neighbouring particles to form new hybrid candidates (mixing depth, normal, rotation and translation). Our motivation in doing so is illustrated by the cube in Figure 6.6; although every corresponding pair of points on the cube from one view to the other has the same $R, \mathbf{t}$, the correct normal depends on the face.

(iii) Random Search. For sampling near a given set of parameters $\mathbf{u}_s$, we first draw a random new state $\mathbf{u}'_s$, and generate the final new sample $\mathbf{u}''_s$ by interpolating $\mathbf{u}_s$ and $\mathbf{u}'_s$ such that $\mathbf{u}''_s = (1-t)\mathbf{u}_s + t\mathbf{u}'_s$, with t ranging from 1 to 0. This mimics the concentric circle search used in the original PatchMatch algorithm, as we decrease the value of t by a factor of 2 after each trial. The normal, depth and translation components are interpolated linearly, and the rotation matrix is interpolated using quaternions. Additionally, we use another strategy to generate new candidates which consists in partially randomising the states, and is part of a joint work and subsequently described in Appendix A.

(iv) RANSAC Plane Fit. This step consists in generating new candidate states by fitting planes to surrounding points using the RANSAC algorithm. It is part of a joint work, and is detailed in Appendix A.

(v) View Propagation. Analogously to [Bleyer et al., 2011, Besse et al., 2012], we use a view propagation step that consists in propagating states from one view to the other if flow is being computed bidirectionally. Since our parameterisation lends itself to a geometric interpretation, view propagation amounts to inverting the homography. In practice, we define $\mathbf{u}'_s = (z'_s, \mathbf{n}'_s, R'_s, \mathbf{t}'_s)^\top$ as the inverted state of $\mathbf{u}_s = (z_s, \mathbf{n}_s, R_s, \mathbf{t}_s)^\top$, where $R'_s = R_s^{-1}$, $\mathbf{t}'_s = -R_s^{-1}\mathbf{t}$, $\mathbf{n}'_s = R_s\mathbf{n}_s$, and z'_s is the depth of the transformed point $z_s R_s \mathbf{p}_s + \mathbf{t}_s$. We parallelise the bi-directional flow computation in order to improve runtime.

6.2.4 Post-processing

It is typically difficult to obtain a good flow at occluded and disoccluded pixels because of the lack of matching data in one of the two images. To account for these areas, we initially compute both the "forward flow" parameters $\mathbf{u}_s^F$ from image 1 to 2, and "backward flow" $\mathbf{u}_s^B$ from image 2 to 1, at each

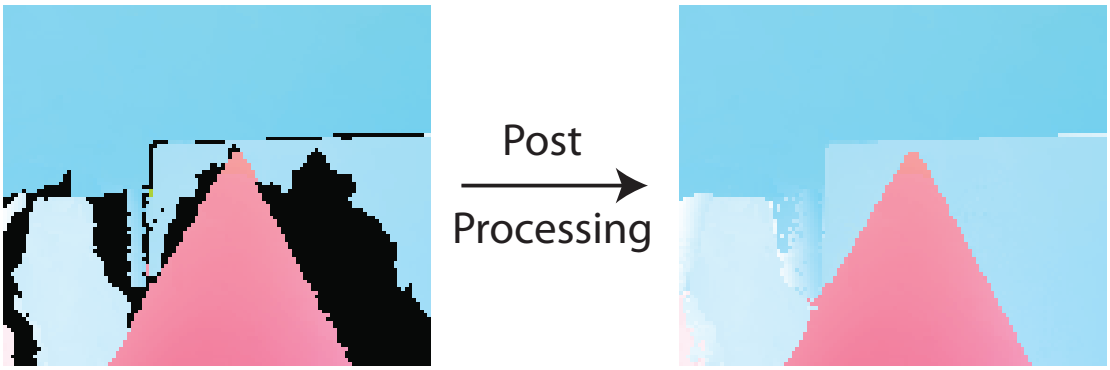


Figure 6.3: Effect of our post processing. Invalid pixels (occluded/disoccluded) are shown in black on the left image.

pixel. Inconsistency between the flow vectors is then measured using the forward-backward error

$$\Delta_s = \|\mathbf{x}_s - H(\mathbf{u}_s^B) * (H(\mathbf{u}_s^F) * \mathbf{x}_s)\|. \tag{6.13}$$

We then use a forward-backward consistency check, marking a node s as inconsistent if Δ_s is greater than 1 pixel.

Inconsistent nodes are often located in those problematic areas, and need to be post processed in order to get a final flow free of artefacts. To do so, for each invalid node we create a list of valid pixels by searching along the four cardinal directions. After that, we select the candidate that is most similar in appearance to the original invalid pixel and copy its homography. Propagating homographies to invalid pixels effectively assumes that those pixels are on the same planar surfaces as the neighbouring valid ones. Using homographies instead of 2D displacement has the additional advantage of enabling us to propagate flow values to the occluded and disoccluded areas in agreement with the supporting planes of the neighbouring valid pixels (not necessarily fronto-parallel). We apply a median filter to generate the final result. Figure 6.3 shows the effect of our post-processing step on occluded and disoccluded pixels.

6.3 Results

With our geometric parameterisation, our algorithm can handle large displacement cases in a natural way, without the need of differentiating between them and those with smaller motions. To illustrate this we have evaluated our method on a number of test cases exhibiting large flow. In particular, we run it on the UCL optical flow dataset [Mac Aodha et al., 2012], which contains both small and large motions (we define the threshold for large motion at 25 pixels). Results can be seen in Table 6.1. We compare our algorithm to four other methods: TV [Zach et al., 2007], LD [Brox et al., 2009], CN [Sun et al., 2010a] and MDP [Xu et al., 2012b]. Our method performs particularly well on the large displacement cases of this dataset, and produces reasonable result for smaller displacements. Quantitative results show that our technique outperforms all 4 other methods in approximately 40% of the cases (60% of the cases for large motion), while the end point error is lower than that of TV and LD in most of the cases. The colour scheme used in Table 6.1 indicates that our approach is the one that is most frequently ranked in the first two positions (in 62.5% of the cases), when compared to the other 4 techniques. A visualisation

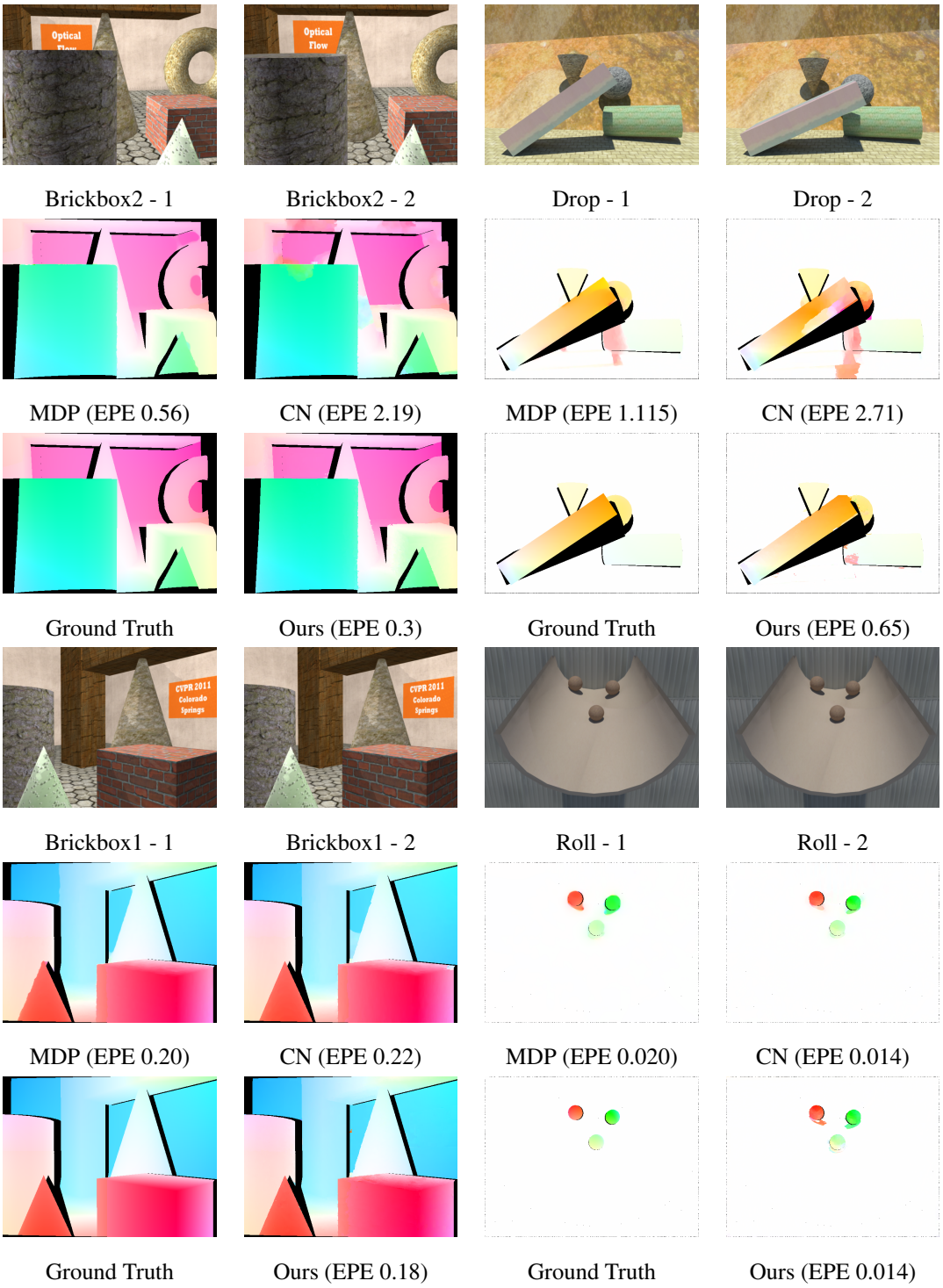
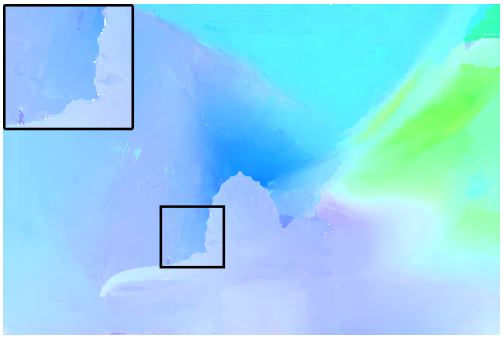


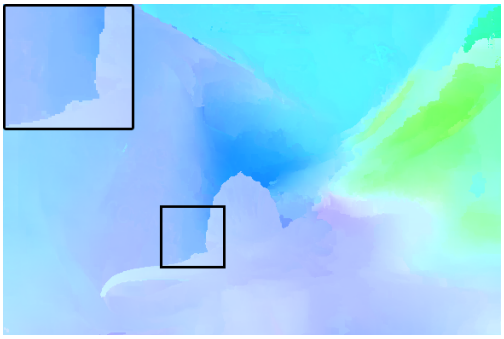
Figure 6.4: Comparison of scenes from the UCL dataset. EPE = End Point Error. CN = Secrets of Optical Flow [Sun et al., 2010a]. MDP = Motion Detail Preserving Optical Flow [Xu et al., 2012b].

(a) Image 1

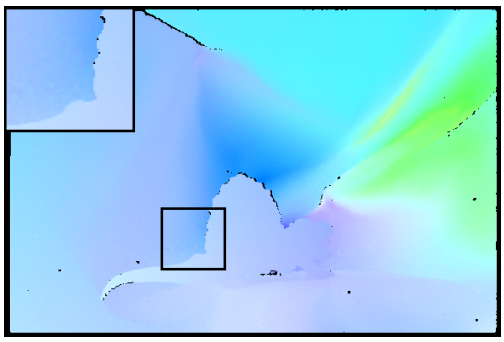
(b) Image 2



(c) No smoothness term ($\lambda = 0$). End point error: 0.159.



(d) Smoothness term ($\lambda = 0.01$). End point error: 0.152.



(e) Smoothness term ($\lambda = 0.01$). End point error: 0.152.

Figure 6.5: Visual comparison illustrating the effect of enabling the pairwise term in our algorithm. (a) and (b) Input images. (c) No smoothness term ($\lambda = 0$). EPE: 0.159. (d) With smoothness term ($\lambda = 0.01$). EPE: 0.149. (e) Ground truth.

	TV	LD	CN	MDP	Ours
UCL Lg. Dis.					
Crates1	3.464	3.10	3.15	**1.65**	2.59
Crates2	4.615	2.51	10.4	**1.35**	1.99
Mayan1	2.331	5.56	1.71	0.48	**0.2**
Robot	2.335	1.212	1.525	**0.7**	1.44
Crates1	1.106	0.54	1.64	0.28	**0.251**
Crates2	3.128	0.809	8.8	**0.37**	0.39
Brickbox1	1.094	2.602	0.22	0.20	**0.16**
Brickbox2	7.478	3.505	2.19	0.56	**0.31**
GrassSky0	2.102	1.039	1.3	0.47	**0.25**
GrassSky9	0.722	0.510	**0.27**	0.29	0.28
blow19	0.525	0.319	0.19	0.26	**0.157**
drop9	5.195	4.369	2.71	1.15	**0.66**
street1	3.647	2.664	4.09	3.19	**1.06**
UCL Sm. Dis.					
Mayan2	0.442	0.35	0.21	0.23	**0.16**
Yosemite	0.310	**0.18**	0.23	3.79	0.44
Grove	0.576	0.48	**0.23**	0.43	0.32
Sponza1	1.006	**0.91**	1.1	1.08	2.54
Sponza2	0.531	**0.48**	1.6	1.77	2.49
TxtR	3.166	0.356	**0.13**	0.19	0.9
TxtL	1.521	0.604	**0.12**	0.23	0.99
blow1Txtr1	0.085	0.081	**0.027**	0.047	0.038
drop1Txtr1	0.119	0.084	0.052	0.057	**0.051**
roll1Txtr1	0.004	0.002	0.002	0.002	**0.001**
roll9Txtr2	0.040	0.023	**0.014**	0.02	**0.014**
Middlebury					
Venus	0.408	0.433	**0.229**	0.28	0.3
Urban3	1.132	0.600	**0.377**	0.42	1.13
Urban2	0.506	0.334	**0.207**	0.32	0.31
RubberWhale	0.135	0.120	**0.077**	0.09	0.11
Hydrangea	0.196	0.178	**0.154**	0.164	0.189
Grove3	0.745	0.657	**0.438**	0.53	0.52
Grove2	0.220	0.149	**0.091**	0.15	0.185
Dimetrodon	0.211	0.117	**0.115**	0.153	0.149

Table 6.1: End point error comparison. TV = A Duality Based Approach for Realtime TV-L1 Optical Flow [Zach et al., 2007]. LD = Large Displacement Optical Flow [Brox et al., 2009]. CN = Secrets of Optical Flow [Sun et al., 2010a]. MDP = Motion Detail Preserving Optical Flow [Xu et al., 2012b]. Colours indicate ranking amongst the 5 methods, from best to worst: green, light green, yellow, orange, red.

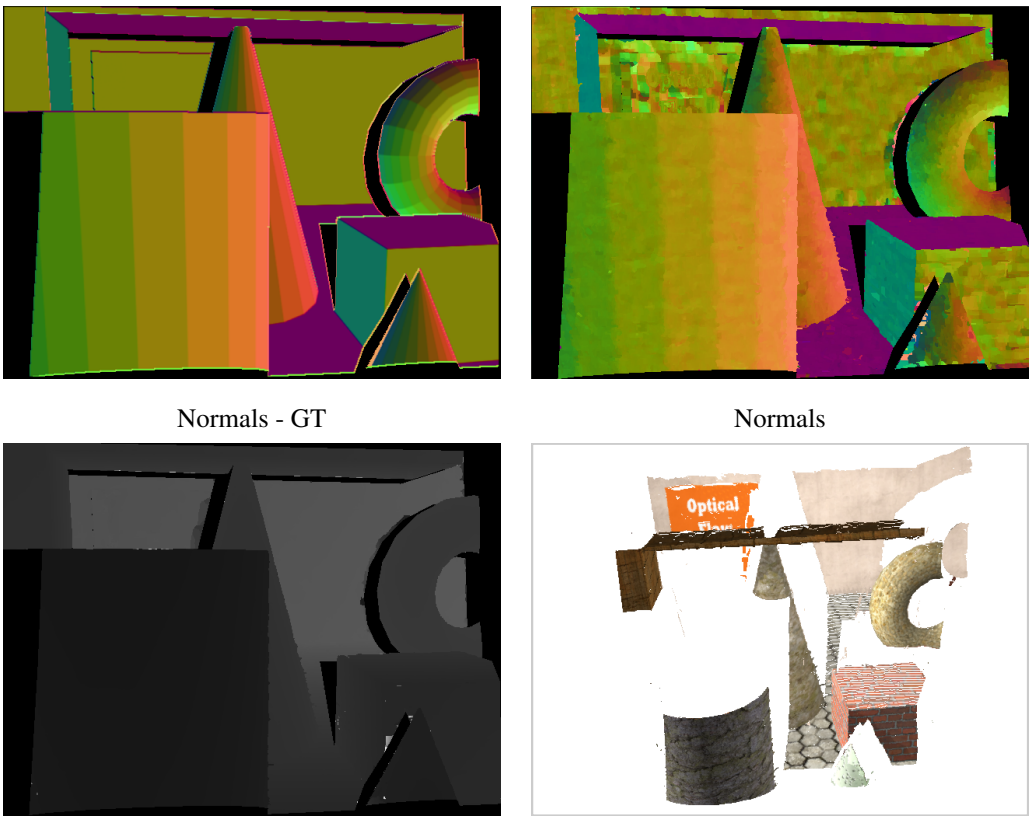


Figure 6.6: Restriction to static scene. Estimation of the plane normals and depth on a static scene, and rendering as a point cloud.

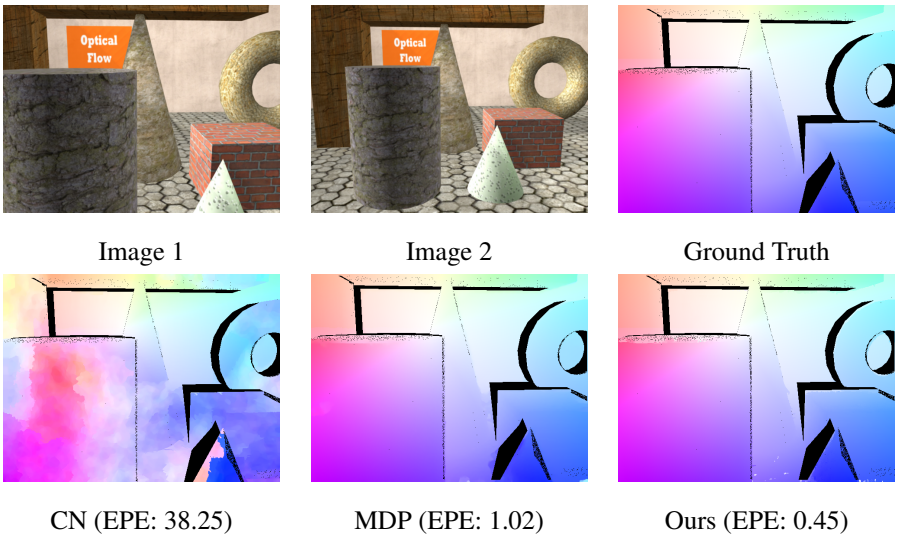


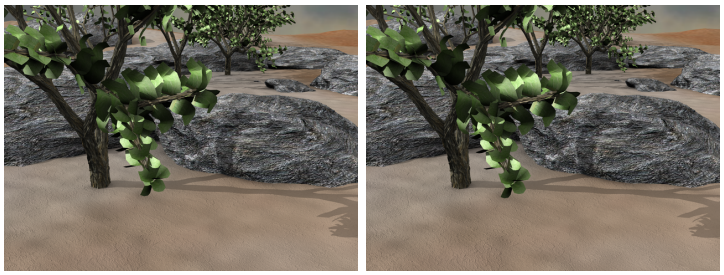
Figure 6.7: Result on a challenging case with camera zoom. Note: this sequence is not part of Table 6.1.

of the produced flow for four cases can be seen in Figure 6.4. The effect of the pairwise term can be seen in Figure 6.5, where we compare running our algorithm with $\lambda = 0$ (no smoothness term) and $\lambda = 0.01$. We use a single particle only, as for a given time budget we did not notice any improvement in using multiple particles. In terms of performance, the average runtime of our experiments was 25 minutes per image pair (640×480) with a patch size of 21×21 on 2 threads (forward and backward directions), which can potentially be further parallelised using the parallel tiled scheme presented in [Barnes et al., 2010].

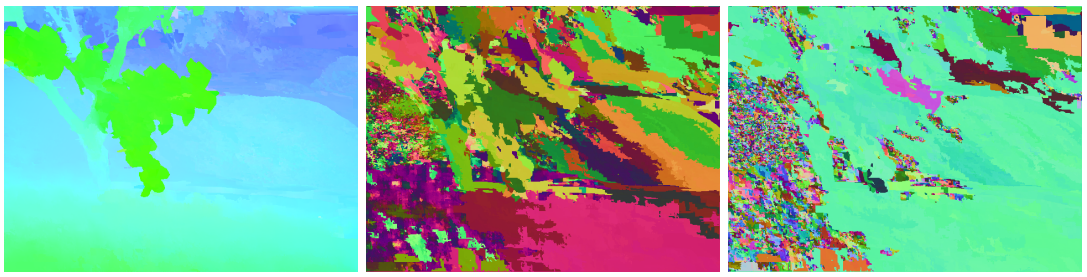
Static Scenes. For static scenes, we force for each pixel $\mathbf{x}_s$ the components R_s and $\mathbf{t}_s$ to be equal to the estimated R_E and $\mathbf{t}_E$ respectively. This has the effect of reducing the parameter space to a 3D space, as it constrains the search to the epipolar lines. While the depth z_s affects the position of the transformed point along its epipolar line, the normal $\mathbf{n}_s$ affects the shape of the warped patch. If there is enough camera motion, we yield a reasonable estimate of the normal fields, as seen in Figure 6.6. On the other hand, little camera motion introduces ambiguity as varying the normals in this case has little effect on the shape of the warped patch. Alongside normals, we also retrieve the depth of each point, which enables us to reconstruct a point cloud of the scene, as seen in Figure 6.6.

Challenging Camera Motions. Certain types of camera motions can be difficult to handle for flow methods that use a 2D parametrisation, such as camera zoom, as it induces a radial flow pattern around the viewing direction, which conflicts with the usual smoothness assumption. Our algorithm however, due to the higher level parametrisation, is flexible enough to recover the homographies induced by this motion as illustrated in Figure 6.7.

Limitations. Contrary to the majority of the optical flow algorithms, we use a multi-scale scheme only for initialisation purposes. This allows us to handle large displacements of smaller objects in a natural way. However, as we do not use the solutions at the down-sampled levels to constrain the optimisation, large textureless areas become more ambiguous. This can be seen in Table 6.1, where most of the high



(a) Image 1 (b) Image 2



(c) Estimated flow (d) Estimated normals (e) Estimated camera translation

Figure 6.8: Illustration of the inability to retrieve meaningful geometric parameters. While matching the two images shown in (a) and (b), we get a reasonable flow shown in (c). However, the normals and camera translations, represented in (d) and (e) with an arbitrary colour scheme, seem to take wrong and non-coherent values.

errors on the datasets “Robot”, “Sponza1”, “Sponza2”, “TxtRMovement”, “TxtLMovement” are due to such areas present in the image. We attribute these failure to the use of only 1 particle, which, as we have seen in chapter 3, is a difficult setting for states to propagate to textureless areas.

Finally, we inspect the 9D parameters of a typical output on a scene that does not contain too much motion, which can be seen in figure 6.8. We observe that the colour maps representing each parameter seems to be broken down into very different areas. This means that our algorithm was unable to recover a unique solution corresponding to the real geometry of the scene, even on the areas of the image corresponding to a single surface. We attribute this failure to the lack of uniqueness of the deformation given by different possible configurations. In other words, several different parameter configurations may yield different homographies which, when applied on a particular patch in the image, seem to have the same deforming effect. Because of the ill-posed nature of the problem, is it difficult to recover meaningful scene geometry when optimising in such a high dimensional space. Further constraints might be needed to recover such information, which we have seen when fixing the camera pose when dealing with a static scene, as depicted in figure 6.6 where the plane normals and depth were successfully recovered.

6.4 Discussion

We have presented a new optical flow method that uses a simple and geometrically meaningful model. The simplicity of our method lies with the geometrical formulation, which makes it flexible to use with intuitive unary and pairwise terms. One of our contributions is to show that it is in possible to optimise

on a such high dimensional space, by using PMBP. We obtain a 2D flow comparable to other state-of-the-art techniques and naturally handle small and large displacements using the same settings. Scenes with large rigid components can benefit from an epipolar geometry-aware prior at those pixels where it is appropriate, and the equivalent of a higher-order smoothness prior elsewhere. Finally, as a byproduct of our approach, depth can be directly extracted from the parameterisation for these rigid components which can then be used to build a point cloud.

However, the problem that we are trying to solve is ill-posed, and we observed that while we obtain good quality optical flow, the 9D parameters of the final solution can be ambiguous and do not always accurately represent the true geometry of the scene. In fact, several different configurations for a given patch seem to produce a similar deformation, which is expected due to the nature of the problem.

This can be potentially solved in future work by adding more constraints. For example, one could use RGBD data, which includes depth information as well as pixel colours and can be captured with devices such as the Kinect [Izadi et al., 2011]. With this additional input, it might be easier to make assumptions about plane normals, and the dimensionality of the search space is reduced. Future work also includes adding motion clustering, where the main motion discovery process that we performed using SIFT features could be further extended to multiple motions. Each motion would represent a cluster, and the optimisation could randomly assign clusters to each pixel while penalising any deviation from these clusters. Cluster refinement could then be run after each iteration, using the RANSAC algorithm once again as we have access to dense pixel motions over the whole image, which is the optical flow that our algorithm is estimating.

Chapter 7

Conclusion

7.1 Summary and conclusions

In this thesis we have presented a novel algorithm named PatchMatch Belief Propagation and we showed its benefits when applied to various correspondence applications. The core of this algorithm has been described in detail in Chapter 3. The next chapters described various applications that use PMBP as the optimisation algorithm. In Chapter 4, we described a stereo matching application based on PatchMatch stereo [Bleyer et al., 2011] extended with PMBP and a novel pairwise term. In Chapter 5 we investigated whether PatchMatch is a suitable optimiser for optical flow, where we design a 5D model which we apply to fast video editing. We then proposed two models using PMBP. The first is an extension of the 5D model with an adapted pairwise term, while the second is a 6D affine model. Finally, Chapter 6 presented a highly over-parameterised model using homography built from a geometrical model and making use of the epipolar geometry of the scene.

7.1.1 PatchMatch Belief Propagation

In Chapter 3, we presented our core optimisation method, PMBP. We started by exposing the link between two existing algorithms: PatchMatch and Belief Propagation, which is our first contribution. PatchMatch is a fast nearest neighbour search algorithm which takes advantage of the coherence of natural images. Through a simple propagation step, combined with a randomised step, it is a powerful optimiser for unary energies. However, it lacks an explicit smoothness term. Belief Propagation on the other hand, is able to optimise energies including a pairwise term, but is very computationally expensive, which limits its applicability to tasks involving large and continuous search spaces. Our algorithm, and second contribution, is called PMBP, and it combines the strong points of both approach while addressing their weaknesses. We use a Particle Belief Propagation formulation as our base algorithm, and add the mechanisms of PatchMatch, which we express under the same framework. The result is an efficient method that can be used on large and continuous search spaces while including an explicit smoothness term.

We validated the performance of PMBP through a series of experiments. After analysing the different steps of PatchMatch, we showed that for an application such as denoising, PatchMatch optimises an energy that only includes a unary term, which can cause the error to increase after a point, where

it starts matching the noise. In comparison, we showed that PMBP is able to provide better results as the energy also comprises a pairwise term. We then compared our algorithm with a discretised version of Belief Propagation and showed that it benefits from the particle resampling, which allows us to efficiently find solutions in the continuous space, which is a benefit over standard BP. We also compared the performance of PMBP with Particle Belief Propagation, and observed that our algorithm is orders of magnitude faster, mainly due to the propagation step that we added. Finally, we ran a further analysis of PMBP by varying various crucial parameters. We showed that the size of the search space is one of the most influential factors on the results. An additional factor is the use of multiple particles that allows us to reach a lower energy, at the cost of increased computational time.

We also exposed some of the weaknesses that appear when using particular settings, notably when the number of particles used is low. In that case, PMBP cannot cope with large textureless areas. This is caused by what we call "pairwise-miss", where the lack of variety in the particle positions of large coherent areas gives low confidence to new and potentially correct candidates. This is a problem that could be addressed in future work, as discussed later in this chapter. The following sections summarise the conclusions of various applications using PMBP.

7.1.2 Stereo Matching

Chapter 4 presented our stereo matching application based on PMBP. It extends the recent PatchMatch stereo method [Bleyer et al., 2011], which over-parameterises the search space and attempts at retrieving a disparity plane at each pixel. The use of such planes is well suited to the propagation step of PatchMatch, and in turn PMBP, as the disparity of all the pixels on a same plane in the scene can be explained by a unique state representing this plane, which can be propagated to neighbouring nodes. Furthermore, this parameterisation is well adapted to the randomisation step of PatchMatch and PMBP, as a single good initial guess is adequate to find a reasonable solution for all the pixels on a planar surface. We designed a pairwise term that can be applied to such planes, and encourages neighbouring pixels to have similar supporting planes.

Our experiments validated the benefits of PMBP over PatchMatch on the stereo matching application. We first showed that incorporating a pairwise term in our model is always beneficial for a wide variety of scenes. To do so we gradually increased the smoothness level on a number of cases, and observed the impact and benefits of our optimiser which returned lower errors, both for raw and post-processed results. We noted that our algorithm is a generalisation of PatchMatch by can easily be reproduced by simply setting the smoothness level to 0. To further validate our approach we also showed state-of-the-art results on the public Middlebury dataset, where we ranked first on the sub-pixel accuracy level at time of publication of our paper [Besse et al., 2012], which won the best industrial impact award at BMVC 2012. We analysed the effects of the post-processing step which enable the correction of occluded and disoccluded areas in the image. Finally, we ran experiments to show the benefits of the view-propagation step, that further improves the results as is it a source of good candidate particles.

Limitations of our stereo algorithm include the high number of parameters which might require extensive tuning in order to be able to provide good results on a large variety of cases. Also, the cur-

rent implementation does does not allow real-time applications, due to the over-parameterisation which involves expensive computations in order to output the disparity. Finally, the weaknesses of PMBP on large and textureless areas are naturally present in this application when using low number of particles as mentioned in the previous section.

7.1.3 Over-parameterised Optical Flow

In Chapter 5, we applied our algorithm to optical flow computations. Here we over-parameterised the state space in different ways. We first showed that using PatchMatch for optical flow is an efficient optimiser providing reasonable results. To our our knowledge there is no record in the literature of such attempt, except from [Boltz and Nielsen, 2010] who combine super-pixels with PatchMatch to output a dense flow estimation of the scene. We used a 5D model that includes patch translation, rotation and anisotropic scaling. We designed a fast video editing application, consisting in propagating efficiently the edits of a user made on the first frame to the rest of the video. We show that we can obtain reasonable video editing results on various types of surfaces.

The same technique is then optimised with PMBP, instead of PatchMatch, where we designed an adapted pairwise term which uses the displacement of three points of each patch to add smoothness to the solution. We validate the benefits of PMBP over PatchMatch by using standard optical flow datasets (Middlebury and UCL) where we observe a systematic improvement in the resulting EPE. We also varied the level of smoothness and observed that the minimum error was located at a non-null smoothness value, which corroborates our findings in Chapter 4.

Finally, we further over-parameterise the search space by using an affine model which includes 6 degrees of freedom. This showed that this model can cope better with large deformations, such as those due to perspective distortion. However we observed that using the affine model does induce a slight loss in quality when the data is not complex enough to benefit from the increase in dimensionality. This behaviour is expected as when adding an extra degree of freedom it is more difficult to sample the search space, which leads to a decrease in robustness. We concluded that a flow algorithm using our framework needs to be complex enough to capture the most large deformations, but with as few degrees of freedom as possible in order to ensure robustness. This lead us to consider a geometrical based optical flow, presented in Chapter 6.

7.1.4 Geometrically-based Optical Flow

Finally, in Chapter 6, we propose a novel way of performing optical flow computation by using a geometrical model, with a highly over-parameterised 9 dimensional state space. We use the geometric properties of images captured from pinpoint cameras and propose to recover for each pixel the homography under which it is transformed onto the other view. Due to the fact that the pixels of a plane captured by two different cameras can be related with a single homography, this model is well suited to the randomisation and propagation steps of PMBP, similarly to the stereo model of Chapter 4. Each homography is broken down into subcomponents, which are a depth, a plane normal, and the rotation and translation of the camera. With this formulation, we are able to constrain the search space and perform random search in a more intuitive way, while naturally handling the smooth flow variation on slanted surfaces. One of

our contribution is to show that it is in fact possible to optimise in such high-dimensional space, due to the efficiency of PMBP. Furthermore, using a pre-processing step on the images to extract the main motion of the camera can provide us with a good initialisation of the solution. Our algorithm is simple and flexible, and does not use a highly complex smoothness term. We present a post-processing step that is inspired from that of Chapter 4, but uses similarity in appearance to make the selection of the closest valid pixel from which we copy the final state.

We compare our method to other recent techniques and show that our model achieves very competitive results for large displacements and performs reasonably well for smaller ones. Moreover it can cope with types of deformation that are usually problematic with, such as a camera motion towards the Z axis. However, as the task of finding the geometrical configuration of the scene from only a pair of images is very ill-posed, our method is not always able to recover meaningful parameters, although it produces a valid 2D flow. Furthermore, as we do not use a multi-scale scheme, our algorithm cannot cope with large textureless areas due to the weakness of PMBP described in Chapter 3 when using a low number of particles. A final drawback concerns the computational time, which is rather high, due to the dimensionality of the proposed model.

7.2 Future Work

In this section we describe the potential future work that would either improve or extend to different directions the various algorithms described in this thesis.

Address the weaknesses of PMBP when using low number of particles As explained in Chapter 3, PMBP has difficulties propagating a good solution in large textureless areas when using a low number of particles. This behaviour can be avoided by increasing the number of particles used, however this also increases the computation time. One potential solution to address this problem, while benefiting from the efficiency that a low number of particles provides, is part of future work and could consists in introducing a background value to all the distributions used. In the probabilistic framework, this means that each message and belief is represented as a set of particles, as well as a constant background value. This background value depends on the pixel of the image and would typically be higher for pixels belonging to textureless areas, making the values at the particles less important. This way, the propagation would be more "tolerant" for these pixels, and coherent regions set with a wrong solution would be more easily replaced by correct states.

Enforce particle diversity in PMBP PatchMatch uses a "keep k-best" scheme while resampling particles. This can often result to a set of particles that contains very similar states, which is not desirable in certain cases, such as during the propagation step, over the course of several iterations. There, a state might be proposed repeatedly by the different neighbouring nodes, which might cause this states to enter several times the particle set. Future work includes addressing this problem and finding a way of enforcing a certain amount of diversity in the particle set.

Apply our stereo algorithm to videos PMBP seems to be well suited for performing correspondence field estimation over consecutive frames of a video. Indeed, a temporal smoothness of the solution

is generally sought in order to avoid flickering artefacts from one frame to an other. In particular, [Bleyer et al., 2011] showed good performance when applying their stereo algorithm to videos. As our method is related to theirs, future work naturally includes applying PMBP to a sequence of images while designing a temporal pairwise term that will encourage the solution to be smooth in this extra dimension.

Improve the parameters used in our various applications PMBP and in particular the different energies that we used throughout this thesis comprise a rather large number of parameters, which can complexify the method, especially from a user's point of view. In our work, we used values that were chosen either empirically, or based on limited experimental work. A possibility of improving this point is to reduce the number of parameters by finding simpler - but equivalent - models. Another option would be to learn the optimal parameters using a large-scale training dataset, to provide the user with preset parameters that are known to generate sensible outputs.

Add motion clustering to our geometrically based optical flow model In our geometrically based optical flow, we initially extract SIFT features and run a RANSAC algorithm to estimate the main motion of the camera, and use this information as part as our initialisation. While this approach is particularly beneficial for the background objects, it might not be sufficient for cases with more than one motion in the scene, such as a camera motion in addition to multiple independent object motions. To overcome this issue, future work could include a motion clustering process to our method. This involves detecting more than one motion initially, and define clusters which are pixels that agree to the same camera motion. The clusters can then be refined as the optical flow algorithm evolves, as it gives us a dense pixel correspondence which can then be used again with a RANSAC algorithm. Clustering would also benefit from an appropriate pairwise term which would encourage all the pixels of a same cluster to follow the same camera motion. A challenge would be to accurately detect small clusters, corresponding for example to small moving objects.

Apply our geometrically based optical flow model to RGBD data We have seen in Chapter 6 that is is difficult to interpret the extra information returned by our algorithm such as depth, plane normal and camera motion due to the ill-posed nature of the problem. Future work includes applying our algorithm to RGBD data, which contain depth information in addition to standard pixel colours. This type of data can be captured with devices such as the Kinect [Izadi et al., 2011] and is lately commonly used in research. Having depth information will first reduce the dimensionality of the search space, while providing some clues to the surface normals at each pixel. Our algorithm could then potentially be converted to a scene flow algorithm where the camera motion could be estimated more accurately.

List of publications

- [Besse et al., 2012] Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2012). PMBP: Patchmatch belief propagation for correspondence field estimation. In *British Machine Vision Conference (BMVC), 2012.*

- [Besse et al., 2013b] Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2013b). PMBP: Patchmatch belief propagation for correspondence field estimation. In *International Journal of Computer Vision (IJCV), 2013. (Accepted subject to minor revisions).*

- [Besse et al., 2013a] Besse, F., Hornacek, M., Rother, C., Fitzgibbon, A., and Kautz, J. (2013a). Highly overparameterized optical flow using patchmatch belief propagation. In *Pattern Analysis and Machine Intelligence (PAMI), 2013. IEEE Transactions on. (Submitted).*

Appendices

Appendix A

Joint Work

This appendix describes the details of the joint work of Chapter 6 carried out with Michael Hornacek and submitted in [Besse et al., 2013a].

A.1 Minimisation

In 6.2.3, a few additions need to be described. We modify the original PMBP algorithm with a new proposal adoption scheme, update the random search to consider extra candidates and finally use a local RANSAC plane fit to generate new candidate planes.

A.1.1 Proposal Adoption

We use a variation of the MCMC step mentioned in PMBP. We use a new acceptance criterion, which consists in accepting a new candidate $\mathbf{u}'_s$ in the particle set at pixel $\mathbf{x}_s$ if the following condition is satisfied:

$$B_s(\mathbf{u}'_s) - \eta \cdot \text{rand}(0,1) < B_s(\mathbf{u}_s) \tag{A.1}$$

where η controls the level of relaxation (set to 0.005 in our experiments), B_s is the disbelief computed at pixel $\mathbf{x}_s$ and $\mathbf{u}_s$ is the highest disbelief particle in the current set of $\mathbf{x}_s$. We proceed in this manner because we recognise that, in particular at wide-displacements, the optimisation might encounter a local minimum of the full energy owing to sampling artifacts, perspective distortions, or changes in lighting.

A.1.2 Random Search

In addition to the perturbation of all the components of a state simultaneously, we perturb, at random, one of the components $z_s, \mathbf{n}_s, R_s, \mathbf{t}_s$ of the parameter state $\mathbf{u}_s$ currently assigned to the pixel $\mathbf{x}_s$, while keeping the other components of $\mathbf{u}_s$ *effectively* locked. We mean this in the sense that if we choose to perturb the rotation, we do so as if we were rotating the transformed plane at the point $z_s R_s \mathbf{p}_s + \mathbf{t}_s$; likewise, if we choose to perturb the translation, we do so as if we were translating the point $z_s R_s \mathbf{p}_s + \mathbf{t}_s$. In the case of the depth or normal, we simply leave the other components unchanged. We carry out random search for a fixed number of ever-decreasing search ranges.

A.1.3 RANSAC Plane Fit

If $R_s, \mathbf{t}_s$ are reasonable and if at least parts of the reconstructed depths are already plausible, a sensible move is to attempt to refine $\mathbf{n}_s, z_s$ by fitting a plane to the recovered 3D points corresponding to the neighbours $W(s)$ of $\mathbf{x}_s$ using RANSAC. The candidate normal is simply the normal—constrained to point toward the camera—of this plane, and the candidate depth is obtained by intersecting the plane with the back-projection of $\mathbf{x}_s$.

Bibliography

[Arya and Mount, 1993] Arya, S. and Mount, D. M. (1993). Algorithms for fast vector quantization. In *Data Compression Conference (DCC), 1993.*, pages 381–390. IEEE.

[Baker and Matthews, 2004] Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV), 2004.*, 56(3):221–255.

[Baker et al., 2011] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M. J., and Szeliski, R. (2011). A database and evaluation methodology for optical flow. *International Journal of Computer Vision (IJCV), 2011.*, 92(1):1–31.

[Barnard, 1986] Barnard, S. T. (1986). *A stochastic approach to stereo vision*. Defense Technical Information Center.

[Barnes et al., 2009] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. (2009). Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG), 2009.*, 28(3):24.

[Barnes et al., 2010] Barnes, C., Shechtman, E., Goldman, D., and Finkelstein, A. (2010). The generalized patchmatch correspondence algorithm. *Computer Vision (ECCV), 2010. European Conference on*, pages 29–43.

[Bathe and Wilson, 1976] Bathe, K.-J. and Wilson, E. L. (1976). Numerical methods in finite element analysis.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *Computer Vision (ECCV), 2006. European Conference on*, pages 404–417.

[Besse et al., 2013a] Besse, F., Hornacek, M., Rother, C., Fitzgibbon, A., and Kautz, J. (2013a). Highly overparameterized optical flow using patchmatch belief propagation. In *Pattern Analysis and Machine Intelligence (PAMI), 2013. IEEE Transactions on. (Submitted).*

[Besse et al., 2012] Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2012). Pmbp: Patchmatch belief propagation for correspondence field estimation. In *British Machine Vision Conference (BMVC), 2012.*

[Besse et al., 2013b] Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2013b). Pmbp: Patchmatch belief propagation for correspondence field estimation. In *International Journal of Computer Vision (IJCV), 2013. (Accepted subject to minor revisions).*

[Birchfield and Tomasi, 1999] Birchfield, S. and Tomasi, C. (1999). Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision (IJCV), 1999.*, 35(3):269–293.

[Black and Anandan, 1996] Black, M. J. and Anandan, P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding, 1996.*, 63(1):75–104.

[Bleyer et al., 2011] Bleyer, M., Rhemann, C., and Rother, C. (2011). Patchmatch stereo-stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC), 2011.*

[Boltz and Nielsen, 2010] Boltz, S. and Nielsen, F. (2010). Randomized motion estimation. In *Image Processing (ICIP), 2010. IEEE International Conference on*, pages 781–784. IEEE.

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239.

[Boykov and Jolly, 2001] Boykov, Y. Y. and Jolly, M.-P. (2001). Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision (ICCV), 2001. IEEE International Conference on*, volume 1, pages 105–112. IEEE.

[Brox et al., 2009] Brox, T., Bregler, C., and Malik, J. (2009). Large displacement optical flow. In *Computer Vision and Pattern Recognition (CVPR), 2009. IEEE Conference on*, pages 41–48. IEEE.

[Brox et al., 2004] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. *Computer Vision (ECCV), 2004. European Conference on*, pages 25–36.

[Bruhn et al., 2005] Bruhn, A., Weickert, J., and Schnörr, C. (2005). Lucas/kanade meets horn/schunck: Combining local and global optic flow methods. *International Journal of Computer Vision (IJCV), 2005.*, 61(3):211–231.

[Buades et al., 2005] Buades, A., Coll, B., and Morel, J.-M. (2005). A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR), 2005. IEEE Conference on*, volume 2, pages 60–65. IEEE.

[Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *Computer Vision (ECCV), 2012. European Conference on*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR), 2005. IEEE Conference on*, volume 1, pages 886–893. IEEE.

[Dasgupta and Freund, 2008] Dasgupta, S. and Freund, Y. (2008). Random projection trees and low dimensional manifolds. In *ACM Symposium on Theory of Computing (STOC), 2008.*, pages 537–546. ACM.

[Datar et al., 2004] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry (SOCG), 2004.*, pages 253–262. ACM.

[Decarlo and Metaxas, 2000] Decarlo, D. and Metaxas, D. (2000). Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision (IJCV), 2000.*, 38(2):99–127.

[Felzenszwalb and Huttenlocher, 2006] Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision (IJCV), 2006.*, 70(1):41–54.

[Fletcher, 1976] Fletcher, R. (1976). Conjugate gradient methods for indefinite systems. *Numerical Analysis, 1976.*, pages 73–89.

[Freeman and Adelson, 1991] Freeman, W. T. and Adelson, E. H. (1991). The design and use of steerable filters. *Pattern Analysis and Machine Intelligence (PAMI), 1991. IEEE Transactions on*, 13(9):891–906.

[Friedman et al., 1977] Friedman, J. H., Bentley, J. L., and Finkel, R. A. (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS), 1977.*, 3(3):209–226.

[Geiger et al., 2012] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012. IEEE Conference on*, Providence, USA.

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence (PAMI), 1984. IEEE Transactions on*, (6):721–741.

[HaCohen et al., 2011] HaCohen, Y., Shechtman, E., Goldman, D. B., and Lischinski, D. (2011). Non-rigid dense correspondence with applications for image enhancement. *ACM Transactions on Graphics (TOG), 2011.*, 30(4):70.

[Hannah, 1974] Hannah, M. J. (1974). Computer matching of areas in stereo images.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50. Manchester, UK.

[Hartley and Zisserman, 2000] Hartley, R. and Zisserman, A. (2000). *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press.

[He et al., 2011a] He, K., Rhemann, C., Rother, C., Tang, X., and Sun, J. (2011a). A global sampling method for alpha matting. In *Computer Vision and Pattern Recognition (CVPR), 2011. IEEE Conference on*, pages 2049–2056. IEEE.

[He and Sun, 2012] He, K. and Sun, J. (2012). Computing nearest-neighbor fields via propagation-assisted kd-trees. In *Computer Vision and Pattern Recognition (CVPR), 2012. IEEE Conference on*, pages 111–118. IEEE.

[He et al., 2010] He, K., Sun, J., and Tang, X. (2010). Guided image filtering. *Computer Vision (ECCV), 2010. European Conference on*, pages 1–14.

[He et al., 2011b] He, L., Bleyer, M., and Gelautz, M. (2011b). Object removal by depth-guided inpainting. *Austrian Association for Pattern Recognition (OAGM), 2011.*, 2.

[Hel-Or and Hel-Or, 2005] Hel-Or, Y. and Hel-Or, H. (2005). Real-time pattern matching using projection kernels. *Pattern Analysis and Machine Intelligence (PAMI), 2005. IEEE Transactions on*, 27(9):1430–1445.

[Horn and Brooks, 1986] Horn, B. K. and Brooks, M. J. (1986). The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing, 1986.*, 33(2):174–208.

[Horn and Schunck, 1981] Horn, B. K. and Schunck, B. G. (1981). Determining optical flow. *Artificial Intelligence (AI), 1981.*, 17(1):185–203.

[Hosni et al., 2009] Hosni, A., Bleyer, M., Gelautz, M., and Rhemann, C. (2009). Local stereo matching using geodesic support weights. In *Image Processing (ICIP), 2009. IEEE International Conference on*, pages 2093–2096. IEEE.

[Ihler et al., 2009] Ihler, A. T., Frank, A. J., and Smyth, P. (2009). Particle-based variational inference for continuous systems. NIPS.

[Isard, 2003] Isard, M. (2003). Pampas: Real-valued graphical models for computer vision. In *Computer Vision and Pattern Recognition (CVPR), 2003. IEEE Conference on*, volume 1, pages I–613. IEEE.

[Isard et al., 2009] Isard, M., MacCormick, J., and Achan, K. (2009). Continuously-adaptive discretization for message-passing algorithms. In *Neural Information Processing Systems (NIPS), 2009.*, pages 737–744.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011). Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *ACM Symposium on User Interface Software and Technology (UIST), 2011.*, pages 559–568. ACM.

[Jolliffe, 2005] Jolliffe, I. (2005). *Principal component analysis*. Wiley Online Library.

[Kanade et al., 1995] Kanade, T., Kano, H., Kimura, S., Yoshida, A., and Oda, K. (1995). Development of a video-rate stereo machine. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 95–100. IEEE.

[Ke and Sukthankar, 2004] Ke, Y. and Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition (CVPR), 2004. IEEE Conference on*, volume 2, pages II–506. IEEE.

[Klaudiny and Hilton, 2011] Klaudiny, M. and Hilton, A. (2011). Cooperative patch-based 3d surface tracking. In *Visual Media Production (CVMP), 2011. Conference for*, pages 67–76. IEEE.

[Klaus et al., 2006] Klaus, A., Sormann, M., and Karner, K. (2006). Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 15–18. IEEE.

[Koller et al., 1999] Koller, D., Lerner, U., and Angelov, D. (1999). A general algorithm for approximate inference and its application to hybrid bayes nets. In *Uncertainty in Artificial Intelligence (UAI), 1999. Proceedings of the Fifteenth Conference on*, pages 324–333. Morgan Kaufmann Publishers Inc.

[Kolmogorov and Zabih, 2001] Kolmogorov, V. and Zabih, R. (2001). Computing visual correspondence with occlusions using graph cuts. In *Computer Vision (ICCV), 2001. IEEE International Conference on*, volume 2, pages 508–515. IEEE.

[Komodakis, 2006] Komodakis, N. (2006). Image completion using global optimization. In *Computer Vision and Pattern Recognition (CVPR), 2006. IEEE Conference on*, volume 1, pages 442–452. IEEE.

[Korman and Avidan, 2011] Korman, S. and Avidan, S. (2011). Coherency sensitive hashing. In *Computer Vision (ICCV), 2011. IEEE International Conference on*, pages 1607–1614. IEEE.

[Kothapa et al., 2011] Kothapa, R., Pacheco, J., and Sudderth, E. B. (2011). *Max-product particle belief propagation*. PhD thesis, Masters thesis, Brown University.

[Lee et al., 2010] Lee, K. J., Kwon, D., Yun, I. D., and Lee, S. U. (2010). Optical flow estimation with adaptive convolution kernel prior on discrete framework. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2504–2511. IEEE.

[Lempitsky et al., 2008] Lempitsky, V., Roth, S., and Rother, C. (2008). Fusionflow: Discrete-continuous optimization for optical flow estimation. In *Computer Vision and Pattern Recognition (CVPR), 2008. IEEE Conference on*, pages 1–8. IEEE.

[Lewis, 1995] Lewis, J. (1995). Fast template matching. In *Vision Interface*, volume 95, pages 15–19.

[Liu et al., 2011] Liu, C., Yuen, J., and Torralba, A. (2011). Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence (PAMI), 2011. IEEE Transactions on*, 33(5):978–994.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV), 2004.*, 60(2):91–110.

[Mac Aodha et al., 2010] Mac Aodha, O., Brostow, G. J., and Pollefeys, M. (2010). Segmenting video into classes of algorithm-suitability. In *Computer Vision and Pattern Recognition (CVPR), 2010. IEEE Conference on*, pages 1054–1061. IEEE.

[Mac Aodha et al., 2012] Mac Aodha, O., Humayun, A., Pollefeys, M., and Brostow, G. (2012). Learning a confidence measure for optical flow.

[MacQueen et al., 1967] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967.*, volume 1, page 14. California, USA.

[Maes et al., 1997] Maes, F., Collignon, A., Vandermeulen, D., Marchal, G., and Suetens, P. (1997). Multimodality image registration by maximization of mutual information. *Medical Imaging (T-MI), 1997. IEEE Transactions on*, 16(2):187–198.

[Mansfield et al., 2011] Mansfield, A., Prasad, M., Rother, C., Sharp, T., Kohli, P., and Van Gool, L. (2011). Transforming image completion. In *British Machine Vision Conference (BMVC), 2011.*

[Mikolajczyk and Matas, 2007] Mikolajczyk, K. and Matas, J. (2007). Improving descriptors for fast tree matching by optimal linear projection. In *Computer Vision (ICCV), 2007. IEEE International Conference on*, pages 1–8. IEEE.

[Min et al., 2011] Min, D., Lu, J., and Do, M. N. (2011). A revisit to cost aggregation in stereo matching: How far can we reduce its computational redundancy? In *Computer Vision (ICCV), 2011. IEEE International Conference on*, pages 1567–1574. IEEE.

[Min and Sohn, 2008] Min, D. and Sohn, K. (2008). Cost aggregation and occlusion handling with wls in stereo matching. *Image Processing (ICIP), 2008. IEEE International Conference on*, 17(8):1431–1442.

[Minka et al., 2005] Minka, T. et al. (2005). Divergence measures and message passing. *Microsoft Research Cambridge, Tech. Rep. MSR-TR-2005-173*.

[Mizukami et al., 2012] Mizukami, Y., Okada, K., Nomura, A., Nakanishi, S., and Tadamura, K. (2012). Sub-pixel disparity search for binocular stereo vision. In *Pattern Recognition (ICPR), 2012. International Conference on*, pages 364–367. IEEE.

[More, 1978] More, J. (1978). The levenberg-marquardt algorithm: implementation and theory. *Numerical Analysis, 1978.*, pages 105–116.

[Mühlmann et al., 2002] Mühlmann, K., Maier, D., Hesser, J., and Männer, R. (2002). Calculating dense disparity maps from color stereo images, an efficient implementation. *International Journal of Computer Vision (IJCV), 2002.*, 47(1):79–88.

[Nir et al., 2008] Nir, T., Bruckstein, A. M., and Kimmel, R. (2008). Over-parameterized variational optical flow. *International Journal of Computer Vision (IJCV), 2008.*, 76(2):205–216.

[Noorshams and Wainwright, 2011] Noorshams, N. and Wainwright, M. J. (2011). Stochastic belief propagation: Low-complexity message-passing with guarantees. In *Communication, Control, and Computing (Allerton), 2011. 49th Annual Allerton Conference on*, pages 269–276. IEEE.

[Nowozin and Lampert, 2011] Nowozin, S. and Lampert, C. H. (2011). *Structured learning and prediction in computer vision*, volume 6. Now publishers Inc.

[Omohundro, 1989] Omohundro, S. M. (1989). *Five balltree construction algorithms*. International Computer Science Institute.

[Pal et al., 2006] Pal, C., Sutton, C., and McCallum, A. (2006). Sparse forward-backward using minimum divergence beams for fast training of conditional random fields. In *Acoustics, Speech and Signal Processing (ICASSP), 2006. IEEE International Conference on*, volume 5, pages V–V. IEEE.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann.

[Psarakis and Evangelidis, 2005] Psarakis, E. Z. and Evangelidis, G. D. (2005). An enhanced correlation-based method for stereo correspondence with subpixel accuracy. In *Computer Vision (ICCV) 2005. IEEE International Conference on*, volume 1, pages 907–912. IEEE.

[Rav-Acha et al., 2008] Rav-Acha, A., Kohli, P., Rother, C., and Fitzgibbon, A. (2008). Unwrap mosaics: a new representation for video editing. In *ACM Transactions on Graphics (TOG), 2008.*, volume 27, page 17. ACM.

[Rhemann et al., 2011] Rhemann, C., Hosni, A., Bleyer, M., Rother, C., and Gelautz, M. (2011). Fast cost-volume filtering for visual correspondence and beyond. In *Computer Vision and Pattern Recognition (CVPR), 2011. IEEE Conference on*, pages 3017–3024. IEEE.

[Richardt et al., 2010] Richardt, C., Orr, D., Davies, I., Criminisi, A., and Dodgson, N. (2010). Real-time spatiotemporal stereo matching using the dual-cross-bilateral grid. *Computer Vision (ECCV), 2010. European Conference on*, pages 510–523.

[Rivera and Gould, 2011] Rivera, P. and Gould, S. (2011). Simultaneous multi-class pixel labeling over coherent image sets. In *Digital Image Computing Techniques and Applications (DICTA), 2011. International Conference on*, pages 99–106. IEEE.

[Roth and Black, 2007] Roth, S. and Black, M. J. (2007). On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50.

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena, 1992.*, 60(1):259–268.

[Scharstein, 1994] Scharstein, D. (1994). Matching images by comparing their gradient fields. In *Pattern Recognition (ICPR), 1994. International Conference on*, volume 1, pages 572–575. IEEE.

[Scharstein and Szeliski, 2002] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV), 2002.*, 47(1):7–42.

[Seitz and Baker, 2009] Seitz, S. M. and Baker, S. (2009). Filter flow. In *Computer Vision (ICCV), 2009. IEEE International Conference on*, pages 143–150. IEEE.

[Shahar et al., 2011] Shahar, O., Faktor, A., and Irani, M. (2011). Space-time super-resolution from a single video. In *Computer Vision and Pattern Recognition (CVPR), 2011. IEEE Conference on*, pages 3353–3360. IEEE.

[Silpa-Anan and Hartley, 2008] Silpa-Anan, C. and Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition (CVPR), 2008. IEEE Conference on*, pages 1–8. IEEE.

[Slesareva et al., 2005] Slesareva, N., Bruhn, A., and Weickert, J. (2005). Optic flow goes stereo: A variational method for estimating discontinuity-preserving dense disparity maps. *Pattern Recognition (ICPR), 2005. International Conference on*, pages 33–40.

[Sproull, 1991] Sproull, R. F. (1991). Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica, 1991.*, 6(1):579–589.

[Stefano et al., 2004] Stefano, L. D., Marchionni, M., and Mattoccia, S. (2004). A fast area-based stereo matching algorithm. *Image and Vision Computing, 2004.*, 22(12):983–1005.

[Sudderth et al., 2002] Sudderth, E. B., Ihler, A. T., Freeman, W. T., and Willsky, A. S. (2002). Nonparametric belief propagation and facial appearance estimation.

[Sudderth et al., 2010] Sudderth, E. B., Ihler, A. T., Isard, M., Freeman, W. T., and Willsky, A. S. (2010). Nonparametric belief propagation. *Communications of the ACM, 2010.*, 53(10):95–103.

[Sun et al., 2010a] Sun, D., Roth, S., and Black, M. J. (2010a). Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010. IEEE Conference on*, pages 2432–2439. IEEE.

[Sun et al., 2008] Sun, D., Roth, S., Lewis, J., and Black, M. (2008). Learning optical flow. *Computer Vision (ECCV), 2008. European Conference on*, pages 83–97.

[Sun et al., 2010b] Sun, D., Sudderth, E., and Black, M. (2010b). Layered image motion with explicit occlusions, temporal consistency, and depth ordering. *Advances in Neural Information Processing Systems (NIPS), 2010.*, 23:2226–2234.

[Sun et al., 2003] Sun, J., Zheng, N.-N., and Shum, H.-Y. (2003). Stereo matching using belief propagation. *Pattern Analysis and Machine Intelligence (PAMI), 2003. IEEE Transactions on*, 25(7):787–800.

[Tae-o sot and Nishihara, 2012] Tae-o sot, S. and Nishihara, A. (2012). Iterative gradient-driven patch-based inpainting. *Advances in Image and Video Technology (PSIVT), 2012.*, pages 71–81.

[Tappen and Freeman, 2003] Tappen, M. F. and Freeman, W. T. (2003). Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Computer Vision (ICCV), 2003. IEEE International Conference on*, pages 900–906. IEEE.

[Tikhonov, 1963] Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. In *Soviet Math. Dokl.*, volume 5, pages 1035–1038.

[Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Computer Vision (IJCV), 1998. International Conference on*, pages 839–846. IEEE.

[Trobin et al., 2008] Trobin, W., Pock, T., Cremers, D., and Bischof, H. (2008). An unbiased second-order prior for high-accuracy motion estimation. *Pattern Recognition (ICPR), 2008. International Conference on*, pages 396–405.

[Trucco et al., 2003] Trucco, E., Isgro, F., and Bracchi, F. (2003). Plane detection in disparity space. In *Visual Information Engineering (VIE), 2003. International Conference on*, pages 73–76. IET.

[Valgaerts et al., 2008] Valgaerts, L., Bruhn, A., and Weickert, J. (2008). A variational model for the joint recovery of the fundamental matrix and the optical flow. *Pattern Recognition (ICPR), 2008. International Conference on*, pages 314–324.

[Viola and Wells III, 1997] Viola, P. and Wells III, W. M. (1997). Alignment by maximization of mutual information. *International Journal of Computer Vision (IJCV), 1997.*, 24(2):137–154.

[Wainwright et al., 2005] Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005). A new class of upper bounds on the log partition function. *Information Theory (IT), 2005. IEEE Transactions on*, 51(7):2313–2335.

[Wedel et al., 2009] Wedel, A., Cremers, D., Pock, T., and Bischof, H. (2009). Structure-and motion-adaptive regularization for high accuracy optic flow. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1663–1668. IEEE.

[Wedel et al., 2008] Wedel, A., Pock, T., Braun, J., Franke, U., and Cremers, D. (2008). Duality tv-l1 flow with fundamental matrix prior. In *Image and Vision Computing New Zealand (IVCNZ), 2008*, pages 1–6.

[Wei and Levoy, 2000] Wei, L.-Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer Graphics and Interactive Techniques, 2000*, pages 479–488. ACM Press/Addison-Wesley Publishing Co.

[Wei and Quan, 2005] Wei, Y. and Quan, L. (2005). Asymmetrical occlusion handling using graph cut for multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2005. IEEE Conference on*, volume 2, pages 902–909. IEEE.

[Werlberger et al., 2010] Werlberger, M., Pock, T., and Bischof, H. (2010). Motion estimation with non-local total variation regularization. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2464–2471. IEEE.

[Werlberger et al., 2009] Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H. (2009). Anisotropic huber-l1 optical flow. In *BMVC*, pages 1–11.

[Woodford et al., 2009] Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2009). Global stereo reconstruction under second-order smoothness priors. *Pattern Analysis and Machine Intelligence (PAMI), 2009. IEEE Transactions on*, 31(12):2115–2128.

[Xu et al., 2008] Xu, L., Chen, J., and Jia, J. (2008). A segmentation based variational model for accurate optical flow estimation. In *Computer Vision–ECCV 2008*, pages 671–684. Springer.

[Xu et al., 2012a] Xu, L., Dai, Z., and Jia, J. (2012a). Scale invariant optical flow. *Computer Vision (ECCV), 2012. European Conference on*.

[Xu et al., 2012b] Xu, L., Jia, J., and Matsushita, Y. (2012b). Motion detail preserving optical flow estimation. *Pattern Analysis and Machine Intelligence (PAMI), 2012. IEEE Transactions on*, 34(9):1744–1757.

[Yang et al., 2009] Yang, Q., Wang, L., Yang, R., Stewénius, H., and Nistér, D. (2009). Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. *Pattern Analysis and Machine Intelligence (PAMI), 2009. IEEE Transactions on*, 31(3):492–504.

[Yedidia et al., 2005] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory (IT), 2005. IEEE Transactions on*, 51(7):2282–2312.

[Yianilos, 1993] Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete algorithms (SODA), 1993.*, pages 311–321. Society for Industrial and Applied Mathematics.

[Yoon and Kweon, 2006] Yoon, K.-J. and Kweon, I. S. (2006). Adaptive support-weight approach for correspondence search. *Pattern Analysis and Machine Intelligence (PAMI), 2006. IEEE Transactions on*, 28(4):650–656.

[Young, 2003] Young, D. M. (2003). *Iterative solution of large linear systems*. Dover Publications.

[Zach et al., 2007] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime tv-l 1 optical flow. *Pattern Recognition (ICPR), 2007. International Conference on*, pages 214–223.

[Zitnick et al., 2005] Zitnick, C., Jojic, N., and Kang, S. B. (2005). Consistent segmentation for optical flow estimation. In *Computer Vision (ICCV), 2005. IEEE International Conference on*, volume 2, pages 1308–1315. IEEE.